AD-A011 278

A COMPUTER AIDED STATISTICAL COVARIANCE PROGRAM FOR MISSILE SYSTEM ANALYSIS

James R. Rowland, et al

Oklahoma State University

Prepared for:

Army Missile Command

1 April 1974

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

183036

AD-A011278

OFFICE OF

ENGINEERING RESEARCH

OKLAHOMA STATE UNIVERSITY

A COMPUTER AIDED STATISTICAL

COVARIANCE PROGRAM FOR

MISSILE SYSTEM ANALYSIS

D D C
RECEIVED
JUN 23 1975
D

REPORT

TO

U. S. Army

Missile Command

April 1, 1974
Final Report for
Contract DAAH01-72-C-0672

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA 22151

Approved for public release; distribution unlimited.

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD-A011 278
4. TITLE (and Subtitle) A COMPUTER AIDED STATISTICAL COVARIANCE PROGRAM FOR MISSILE SYSTEM ANALYSIS FINAL REPORT FOR CONTRACT DAAH01-72-C-0672		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James R. Rowland and V. M. Gupta		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Office of Engineering Rsch, Oklahoma State Univ Agriculture and Applied Science Stillwater, Oklahoma 74074		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander, US Army Missile Command Attn: AMSMI-RPR Redstone Arsenal, Alabama 35809		12. REPORT DATE 1 April 1974
		13. NUMBER OF PAGES 195
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DDC RECEIVED JUN 23 1975 D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital computer software package
Noise propagation problems
Propagation of errors due to noise
Large-scale missile systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A combined Monte Carlo-direct covariance algorithm digital computer software package has been developed and tested for determining the effects of noise disturbances on large-scale missile systems. The large-scale system was composed of a 15th-order autopilot, a 4th-order actuator subsystem, a 12th order airframe, and a 2nd-order seeker.

(Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Continued)

This report documents the results of a 2-year development effort under Contract DAAH01-72-C-0672. A basic statistical covariance program involving incremental variations about noise-free operating conditions was developed during the first year to calculate the effects of noise propagation for missile systems up to approximately 25th order. Specific tasks during that period included the development and testing of the basic program, establishing accuracy levels on a typical missile system, establishing tradeoff possibilities for improved program operation, and developing and testing automatic programs to be used with existing digital or hybrid simulations. The basic program was expanded for higher-order systems up to approximately 50th order during the second year. Specific tasks included expanding the basic program, simplifying the program via approximations, developing sequential operations, and establishing final guidelines.

Final Report

for

Contract DAAH01-72-C-0672
with the U. S. Army Missile Command
Redstone Arsenal, Alabama

A COMPUTER AIDED STATISTICAL COVARIANCE PROGRAM FOR MISSILE SYSTEM ANALYSIS

by

James R. Rowland and V. M. Gupta
School of Electrical Engineering

Approved for public release; distribution unlimited.

Office of Engineering Research
Oklahoma State University of
Agriculture and Applied Science
Stillwater, Oklahoma 74074

April 1, 1974

DDC
JUN 1975

SUMMARY

A combined Monte Carlo-direct covariance algorithm digital computer software package has been developed and tested for determining the effects of noise disturbances on large-scale missile systems. The combined software package was applied to a thirty-third order math model of a six degree-of-freedom air defense missile system. The large-scale system was composed of a fifteenth-order (15th) autopilot, a fourth-order (4th) actuator subsystem, a twelfth-order (12th) airframe, and a second-order (2nd) seeker. This final report documents the results of a two-year development effort under Contract DAAH01-72-C-0672, which was initiated on April 1, 1972.

A basic statistical covariance program involving incremental variations about noise-free operating conditions was developed during the first year to calculate the effects of noise propagation for missile systems up to approximately 25th order. Specific tasks during that period included the development and testing of the basic program, establishing accuracy levels on a typical missile system, establishing tradeoff possibilities for improved program operation, and developing and testing automatic programs to be used with existing digital or hybrid simulations. The basic program was expanded for higher-order systems up to approximately 50th order during the second year. Specific tasks included expanding the basic program, simplifying the program via approximations, developing sequential

operations, and establishing final guidelines. All eight of these contract objectives and the associated four milestones were met on schedule.

The expanded program is described in Chapters III and IV of this final report with numerical results for a thirty-third order missile system in Chapter V. In particular, Table IV of Chapter IV indicates that nine new subroutines were added to the existing digital computer program, major changes were made in three other subroutines, and seven of the remaining seventeen subroutines required only minor changes. Several innovations, including an adaptive feature for the calculation of certain coefficient matrix elements, were incorporated into the program development. These have been documented in this final report.

Accuracy levels were established for the direct covariance algorithm by comparing with 25 Monte Carlo simulation runs for the large-scale missile system. Figure 13 indicates that excellent results were obtained for several orders of missile systems by using the direct covariance algorithm. It was also shown that the thirty-third order system exhibited harsh nonlinear characteristics during the launch and terminal modes of a typical flight. Therefore, the Monte Carlo technique was utilized during these modes of operation, and the direct covariance algorithm was used during the large mid-portion of the flight. This combined software package is included in Appendix C.

Tradeoff possibilities with respect to accuracy, computational speed, computing equipment requirements (including storage), and program complexity were examined. It was shown that the RK2 integration formula represented an efficient tradeoff between speed and accuracy for covariance matrix calculations. Simplifying approximations were developed to speed up the operation of the combined software package. Constant coefficients were used to replace slowly-varying elements of the A(t) coefficient matrix. It was shown that during the large mid-portion of the flight, where the direct covariance algorithm was applicable, an important approximation involved the propagation of noise through the seeker relay nonlinearities. Output variance calculations for these relays were achieved from Subroutines SNOISE and DETARA by using the resulting output joint probability density function directly. The harsh nonlinearties encountered during launch and terminal modes could not be handled by this simplified approach. Therefore, Monte Carlo runs were needed for these portions of the flight to supplement direct covariance calculations.

An increased accuracy and a significant savings in computational time are realized for those applications where the direct covariance algorithm may be used over a large portion of the flight. It is shown in Chapter V that input noise levels determine the region in which the direct covariance algorithm is applicable. For the thirty-third order system described in Chapter III with the given noise levels, the combined program operated at approximately twice the speed of 25 Monte Carlo simulations with comparable accuracy. Moreover, the combined program operated at approximately six times the speed of 200 Monte Carlo simulations and over thirty times the speed of 1000 Monte Carlo simulations. Based on both accuracy and computational speed, this combined digital computer software package provides improved capabilities for handling noise propagation in large-scale missile system applications.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Computer software packages have proven to be very useful for the application of sophisticated analysis and design algorithms for industrial problems. Their usefulness in providing powerful results in an easily applied form for the user has led to the development of efficient software packages for large-scale systems. One problem area in which software packages are becoming more popular involves those systems having inherent noise problems resulting from random variations in disturbance inputs and/or system parameters. These random variations result in errors being propagated throughout the large-scale systems. A thorough knowledge of the large-scale system dynamics, statistical properties of dynamical systems, and some simulation experience are necessary for the development of computer software packages for these applications. This final report describes the development and testing of a digital computer software package for determining the propagation of errors due to noise in large-scale missile systems.

Background

Previous work on noise propagation problems has focused on the use of the Monte Carlo technique in which large numbers of runs are ensemble-averaged to obtain statistical results. Primary considerations in the use of this traditional approach are the generation of prespecified statistical inputs and the simulation of dynamical systems. A more modern approach based on computing the state covariance matrix directly has become popular in recent years. This new approach, referred to as the direct covariance algorithm, has been applied for an approximate analysis of large-scale nonlinear systems. The development of a computer software package using the direct covariance algorithm would greatly enhance large-scale system analysis capabilities.

The Monte Carlo method uses repeated sample functions as inputs to the model of a mathematical or physical process. Earlier noise propagation studies by the Monte Carlo method were based on the use of analog noise generators. Due to the fact that these generators were not repetitive, the analog approach became unpopular after the recent development of digital pseudo-random number generators. These generators could be used to generate the same numbers as many times as desired and, thus, ease the work of debugging the simulated program. Large amounts of simulated random data are required for acceptable results. For the digital implementation of the Monte Carlo technique, pseudo-random numbers are either drawn from tables (1) or generated from simple relationships within the computer. For the former case the random numbers must be stored and used whenever required. However, for the latter case Chambers (2), Hull and Dobell (3), MacLaren and Marsaglia (4), and Gelder (5) have developed mixed congruential and multiplicative recurrence formulas for generating pseudo-random numbers. The numbers generated are uniformly distributed on the interval (0,1). The uniformly distributed numbers may be converted into zero-mean, unity-variance, Gaussianly distributed random numbers by an exact closed-form expression developed by Box and Muller (6). An alternate, but approximate, method of converting the uniform sequence to a Gaussian sequence utilizes the Central Limit theorem which states that as the number of statistically independent variables is increased without limit, a Gaussian probability distribution is approached for the sum, regardless of the probability distributions of the various variables.

A direct technique (7-12) has resulted from the error covariance matrix propagation in the Kalman filtering equation (13,14). Though exact for linear time-varying systems, the direct covariance algorithm has also been applied for mildly non-linear systems. For example, this technique has been used by Kuhnel and Sage (15) for sensitivity equations about a nominal flight path due to trajectory initial condition dispersions and random system variations. They used a thirty-third order, six degree-of-freedom homing missile model to illustrate the application to a realistic situation. Kuhnel and Sage used only the adjoint method whereas Irwin and Hung (16) applied both direct and adjoint methods for evaluating the state covariance algorithm for large-scale, nonlinear, dynamical systems. An interval-by-interval linearization procedure has also been proposed (17,18). For nonlinear feedback systems, the direct covariance approach has been used by Brown (19-21) for solving trajectory optimization problems. Using a more accurate algorithm about a nominal trajectory, Clark (22, 23) has developed related results.

Rowland and Holmes (24) have shown that the direct covariance technique is more accurate and faster than the Monte Carlo approach. They demonstrated that the direct covariance algorithm can be applied to mildly nonlinear systems with acceptable results by using linearized incremental equations about the noise-free solution. The objective of this effort was to develop a computer software package for the efficient implementation of the direct covariance algorithm.

Derivation of the Direct Covariance Algorithm

Consider the linear, time-varying, dynamical system represented by the vector differential equation

$$\dot{\underline{x}}(t) = A(t)\underline{x}(t) + B(t)\underline{w}(t) \tag{1.1}$$

where $\underline{x}(t)$ is an n-dimensional state vector, $A(t)$ is an n by n matrix, $B(t)$ is an n by m matrix, and $\underline{w}(t)$ is an m-dimensional input noise vector.

The covariance matrix of the state vector (24,25)* is defined as

$$P(t) \triangleq E\{\underline{x}(t)\underline{x}^T(t)\} \tag{1.2}$$

The elements of the input noise vector are zero-mean white noise processes, and their covariance matrix is represented by

$$E\{\underline{w}(t)\underline{w}^T(\tau)\} = Q_{\underline{w}}(t)\ \delta(t-\tau) \tag{1.3}$$

where $\delta(\cdot)$ is the impulse function. The m by m covariance matrix $Q_{\underline{w}}(t)$ may be time-varying in general.

The covariance matrix $P(t)$ may be determined directly in terms of $A(t)$, $B(t)$, and $Q_{\underline{w}}(t)$ by using $\underline{x}(t)$ in (1.2). The solution of the time-varying, linear differential equation given by (1.1) is

$$\underline{x}(t) = \Phi(t,t_0)\ \underline{x}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\ B(\tau)\ \underline{w}(\tau)d\tau \tag{1.4}$$

Therefore, the covariance matrix of $\underline{x}(t)$ may be calculated as

---

* Reprints of (25) and other selected papers are included in Appendix A.

$$P(t) = E\{\underline{x}(t)\underline{x}^T(t)\}$$

$$= E[\{\Phi(t,t_0)\ \underline{x}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\ B(\tau)\ \underline{w}(\tau)d\tau\}$$

$$\cdot \{\Phi(t,t_0)\underline{x}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\ B(\tau)\ \underline{w}(\tau)d\tau\}^T] \qquad (1.5)$$

Since $\underline{x}(t_0)$ and $\underline{w}(t)$ are uncorrelated for all $t>t_0$,

$$P(t) = E[\Phi(t,t_0)\ \underline{x}(t_0)\ \{\Phi(t,t_0)\ x(t_0)\}^T +$$

$$\int_{t_0}^{t} \int_{t_0}^{t} \Phi(t,\tau_1)\ B(\tau_1)\ \underline{w}(t_1)\{\Phi(t,\tau_2)B(\tau_2)\underline{w}(\tau_2)\}^T d\tau_1 d\tau_2]$$

$$= \Phi(t,t_0)\ E\{\underline{x}(t_0)\underline{x}^T(t_0)\}\ \Phi^T(t,t_0)$$

$$\int_{t_0}^{t} \int_{t_0}^{t} \Phi(t,\tau_1)\ B(\tau_1)\ E\{\underline{w}(\tau_1)\underline{w}^T(\tau_2)\}\ B^T(\tau_2)\ \Phi^T(t,\tau_2)d\tau_1 d\tau_2 \quad (1,6)$$

Using (1.3) and the sifting property of the delta function, (1.6) reduces to

$$P(t) \ = \ \Phi(t,t_0)\ P(t_0)\ \Phi^T(t,t_0) +$$

$$\int_{t_0}^{t} \Phi(t,\tau_1)\ B(\tau_1)\ Q_{\underline{w}}(\tau_1)\ B^T(\tau_1)\ \Phi^T(t,\tau_1)d\tau_1 \qquad (1.7)$$

The integral equation in (1.7) may be expressed more conveniently as a matrix differential equation for $P(t)$. In establishing this form, the state transition matrix $\Phi(t,t_0)$ is identified as the solution of the homogeneous linear differential equation

$$\dot{\Phi}(t,t_0) \ = \ \frac{d}{dt}\Phi(t,t_0) \ = \ A\Phi(t,t_0) \qquad (1.8)$$

with the boundary condition $\Phi(t_0,t_0) = I$. Using the relationship in (1.8) to simplify (1.7) gives

$$\dot{P}(t) = \dot{\Phi}(t,t_0)\, P(t_0)\, \Phi^T(t,t_0) + \Phi(t,t_0)\, P(t_0)\, \dot{\Phi}^T(t,t_0)$$

$$+ \int_{t_0}^{t} \frac{\partial\Phi(t,\tau_1)}{\partial t} B(\tau_1) Q_{\underline{w}}(\tau_1)\, B^T(\tau_1)\, \Phi^T(t,\tau_1) d\tau_1$$

$$+ \int_{t_0}^{t} \Phi(t,\tau_1)\, B(\tau_1)\, Q_{\underline{w}}(\tau_1)\, B^T(\tau_1)\, \frac{\partial\Phi^T(t,\tau_1)}{\partial t} d\tau_1$$

$$+ \Phi(t,t)\, B(t)\, Q_{\underline{w}}(t)\, B^T(t)\, \Phi^T(t,t)$$

$$\dot{P}(t) = A(t)\, [\Phi(t,t_0)\, P(t_0)\, \Phi^T(t,t_0)$$

$$+ \int_{t_0}^{t} \Phi(t,\tau_1)\, B(\tau_1)\, Q_{\underline{w}}(\tau_1)\, B^T(\tau_1)\, \Phi^T(t,\tau_1) d\tau_1]$$

$$+ [\Phi(t,t_0)\, P(t_0)\, \Phi^T(t,t_0)$$

$$+ \int_{t_0}^{t} \Phi(t,\tau_1) B(\tau_1) Q_{\underline{w}}(\tau_1) B^T(\tau_1) \Phi^T(t,\tau_1) d\tau_1]^T A^T(t)$$

$$+ B(t)\, Q_{\underline{w}}(t)\, B^T(t) \tag{1.9}$$

where $\Phi(t,t)$ has been replaced by the identity matrix I. Therefore,

$$\dot{P}(t) = A(t)\, P(t) + P(t)\, A^T(t) + B(t)\, Q_{\underline{w}}(t)\, B^T(t) \tag{1.10}$$

The desired result in (1.10) yields P(t) by solving a set of linear differential equations.

Criteria for Comparison

Since the most efficient technique is sought for the study of noise propagation in large-scale systems, the criteria for comparison between the Monte Carlo technique and the direct covariance algorithm play an important role in selecting the most suitable technique. Some of these criteria are discussed in the following paragraphs.

Information Provided

The primary consideration for choosing a simulation technique is greatly influenced by the information provided by that technique. The Monte Carlo technique provides the complete probability density function associated with random phenomena, whereas the direct covariance technique only gives the variance about the nominal trajectory, which serves as the mean value. In many applications of interest, the mean and variance of selected states is all the information that is required for an acceptable analysis of system behavior.

Accuracy

The next criterion for comparison is the accuracy level provided, which varies with different techniques. The direct covariance algorithm gives exact results for linear systems and may be applied to yield acceptable results for mildly nonlinear systems. On the other hand, the results of 25 to 50 Monte Carlo runs may not provide acceptable accuracy, although a high accuracy may be expected with 1000 Monte Carlo runs (24,25). The step size chosen for integration may be used as a control for the tradeoff between accuracy and computational time.

Computer Storage

The computer software package efficiency may also be judged by the computer storage needed for the application of various techniques. The direct covariance algorithm requires somewhat more storage as compared to the Monte Carlo technique. The amount of additional storage depends upon the order of the system being considered as shown in later chapters.

Computational Time

Another objective of an efficient computer software package is to obtain a computationally fast algorithm. The speed and accuracy may be examined with respect to tradeoff possibilities. For extremely accurate results, the computational time needed may be quite large. By the use of large integration step sizes, the computational speed may be increased. There are many approximate techniques which may be used to reduce the computation time. For example, slowly time-varying coefficients may be replaced by constant coefficients and very small variables and coefficients may be replaced by zero. Moreover, if the order of the system can be reduced, a considerable savings in computer time might be realized.

Program Complexity

The computer software package should be simple so that anyone with only limited simulation experience is able to understand it. Due to the inverse relation of the complexity and computation time, the tradeoff between them is possible. With maximum complexity the computer time may be reduced by as much as a factor of ten in certain applications.

Possibilities of Extension

The general computer software package for the direct covariance algorithm is a fundamental step in the subsequent development of an efficient software package for Kalman filtering as a practical estimation algorithm. Furthermore, many approximate nonlinear filtering algorithms are based on similar considerations.

Outline

Following this introductory chapter, the direct covariance algorithm is extended in Chapter II for application to nonlinear systems. In addition, several Monte Carlo tests are performed to determine a suitable discretization procedure for subsequent use in validating the results of the digital computer software package. The software package development and its application to a large-scale missile system are described in Chapter III. A description of the combined Monte Carlo - direct covariance algorithm software package is provided in Chapter IV. Final numerical results using this software package are presented in Chapter V.

CHAPTER II

DIRECT COVARIANCE ALGORITHM EXTENSIONS AND MONTE CARLO TESTING

This chapter defines the general mathematical system under consideration and extends the direct covariance algorithm for this nonlinear case. Numerical results are presented for a second-order nonlinear system to demonstrate the applicability of the algorithm. Thereafter, the problem of modeling continuous white noise inputs on the digital computer is investigated from a more general viewpoint than considered previously. Three modeling representations are presented and then compared on a second-order system. The best of these discretization procedures is used in subsequent chapters to compare the Monte Carlo technique with the direct covariance algorithm on a thirty-third order math model of a six degree-of-freedom air defense missile system.

Mathematical Formulation

Consider the nonlinear, time-varying, dynamical system represented by the vector differential equation

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{w}, t) \tag{2.1}$$

where $\underline{x}$ is the n-dimensional vector of system variables, $\underline{w}$ is an m-dimensional input noise vector, and t is the independent variable representing time.

The input noise vector $\underline{w}(t)$ has a mean value specified by the m-dimensional vector $\eta_{\underline{w}}(t)$ and a covariance matrix $Q_{\underline{w}}(t)$, which is m by m in dimension. These quantities may be defined mathematically as

$$E\{\underline{w}(t)\} \triangleq \eta_{\underline{w}}(t)$$
$$E\{[\underline{w}(t) - \eta_{\underline{w}}(t)]\,[\underline{w}(\tau) - \eta_{\underline{w}}(\tau)]^T\} \triangleq Q_{\underline{w}}(t)\,\delta(t-\tau) \tag{2.2}$$

where $\delta(\cdot)$ is the impulse function.

The covariance matrix of the state $\underline{x}(t)$ is defined as

$$P(t) \triangleq E\{[\underline{x}(t) - \eta_{\underline{x}}(t)]\,[x(\tau) - \eta_{\underline{x}}(\tau)]^T\} \tag{2.3}$$

where $\eta_{\underline{x}}(t)$ is the mean of $\underline{x}(t)$. The problem is to determine P(t) in terms of the mathematical description of the nonlinear system in (2.1) and the properties of the input noise vector given in (2.2).

An Approximate Covariance Analysis of Nonlinear Systems

The application of the direct covariance algorithm developed in Chapter I to the nonlinear system in (2.1) can be achieved as an approximate analysis. Let $\underline{x}_N(t)$ denote the noise-free nominal trajectory obtained by replacing $\underline{w}(t)$ by $\eta_{\underline{w}}(t)$ in (2.1). It is assumed that the input noise disturbances cause sufficiently small deviations about this nominal solution such that $\eta_{\underline{x}}(t) = \underline{x}_N(t)$. Let these small deviations $\underline{\delta x}(t)$ be defined by

$$\underline{\delta x}(t) \triangleq \underline{x}(t) - \underline{x}_N(t) \tag{2.4}$$

Expanding (2.1) in a Taylor's series about $\underline{x}_N(t)$ yields

$$\underline{\delta \dot{x}}(t) = A(t)\,\underline{\delta x}(t) + B(t)\underline{w}(t) \tag{2.5}$$

where

$$A(t) \triangleq \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\substack{\underline{x}(t) = \underline{x}_N(t) \\ \underline{w}(t) = \underline{n}_{\underline{w}}(t)}}$$

$$B(t) \triangleq \left.\frac{\partial \underline{f}}{\partial \underline{w}}\right|_{\substack{\underline{x}(t) = \underline{x}_N(t) \\ \underline{w}(t) = \underline{n}_{\underline{w}}(t)}} \tag{2.6}$$

The approximation made in (2.5) is that the second and all higher-order terms in $\delta\underline{x}$ are negligible when compared to the linear terms. This approximation is valid if the $\delta\underline{x}$ variations are sufficiently small.

To demonstrate the importance of this approximation, consider the second-order nonlinear system investigated in (24,25). The system is described by

$$\dot{x}_1 = -2x_1 + x_2 + \gamma x_2^2 \text{ sign } (x_2)$$

$$\dot{x}_2 = -x_2 + w(t) \tag{2.7}$$

where w(t) is a zero-mean Gaussian white noise process applied for all $t \geq 0$. Figure 1 shows the results from (24,25) by applying the direct covariance algorithm as the input covariance Q_w was increased from 0.01 to 5. As Q_w was increased, the higher-order δx variations in (2.5) became significant and larger errors were obtained. Therefore, the arbitrary application of the direct covariance algorithm to nonlinear systems with severe nonlinearities and/or extremely high input noise levels must be approached with some caution.

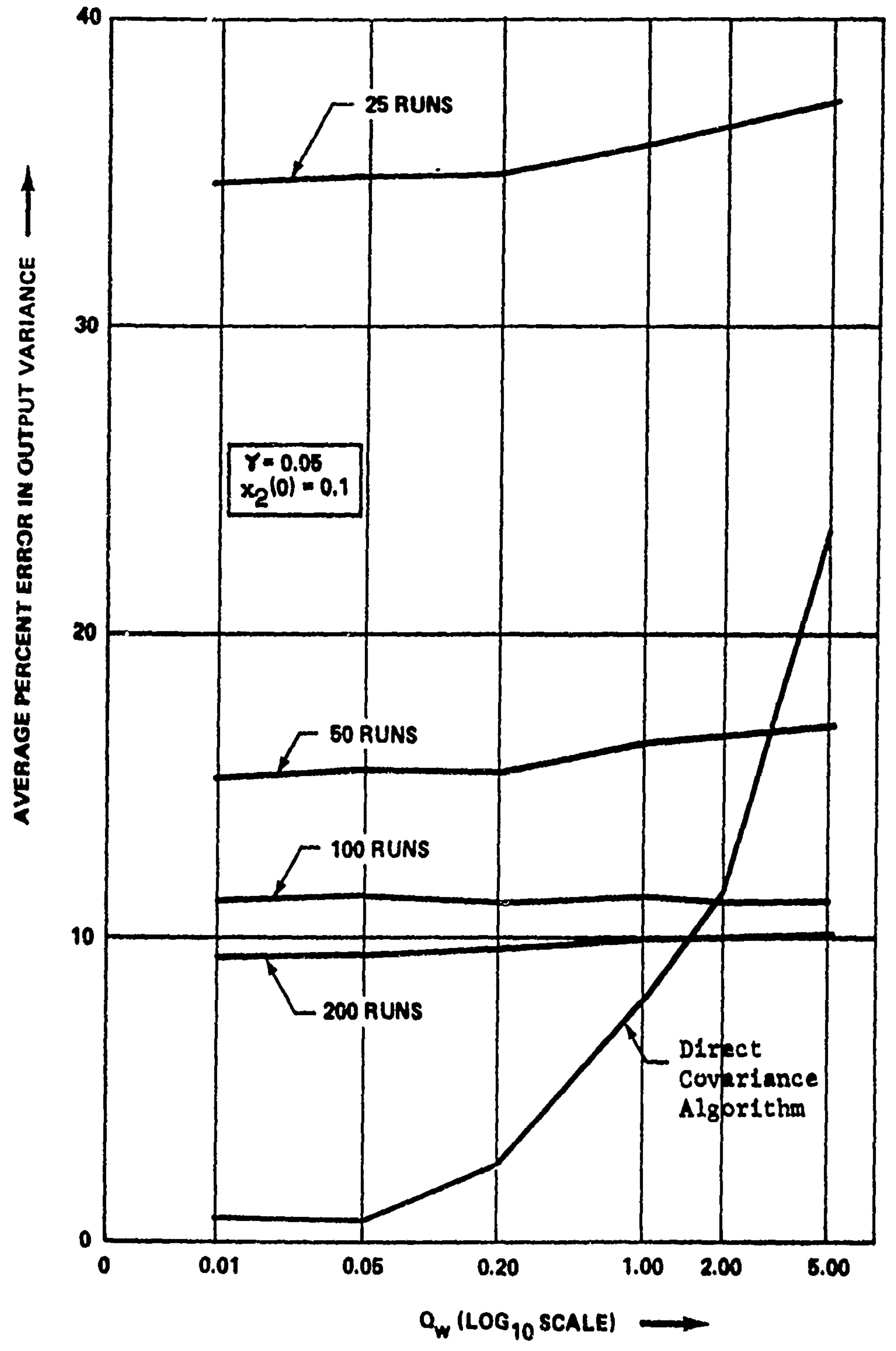


Figure 1. Comparisons Between the Direct Covariance Algorithm and Monte Carlo Simulations for (2.7)

Monte Carlo Testing

To validate the accuracy of the computer software package for the direct covariance algorithm, comparisons were made with the Monte Carlo technique. As a preliminary step, the discretization procedures for white noise inputs were investigated to determine whether improved Monte Carlo results could be obtained. Previous methods were based on the generation of pseudo-random numbers which were then held constant over the discretization interval. The relationships between the covariance matrix $Q_{\underline{w}_d}$ of discrete random sequences and $Q_{\underline{w}}$ defined in (2.2) is given by

$$Q_{\underline{w}_d} = Q_{\underline{w}} / T \tag{2.8}$$

where T is the discretization interval. An extensive study was performed by Rowland and Holmes (24) on the above method, and some of those results are used here to evaluate new methods for the discrete representation of continuous white noise processes.

A new functional approach to the discretization problem has been developed in this work, and results are compared with the previous method in the next section. Suppose several zero-mean random numbers β_k are combined on each discretization interval to form a power series function of time as

$$w_d(\beta_0, \beta_1, \beta_2, \ldots, \beta_K, t) = \sum_{k=0}^{K} \beta_k t^k \quad \text{for } 0 < t < T \tag{2.9}$$

The autocorrelation function of such a train of pulses is given in (26, 27) by

$$R_{w_d w_d}(t, t+\tau) = \begin{cases} \sum_{k=0}^{K} Q_{\beta_k} t^{2k} \left(1 - \frac{|\tau|}{T}\right) & \text{for } |\tau| < T \\ 0 & \text{Otherwise} \end{cases} \tag{2.10}$$

where Q_{β_k} is the variance of β_k. The associated power spectral density is

$$S_{w_d w_d}(\omega) = \int_{-\infty}^{\infty} e^{-j\omega\tau} \left[\underset{T\to\infty}{\text{limit}} \frac{1}{2T} \int_{-T}^{T} R_{w_d w_d}(t,t+\tau)dt\right]d\tau$$

$$= \frac{2(1 - \cos \omega T)}{\omega^2} \sum_{k=0}^{K} Q_{\beta_k} \left(\frac{T^{2k-1}}{2k+1}\right) \quad (2.11)$$

Note that the expression in (2.11) takes advantage of the periodicity of (2.10) and is valid even though the discrete representation of the given continuous random process is nonstationary.

For the continuous white noise case, the autocorrelation function in given by the impulse function

$$R_{ww}(\tau) = Q_w \delta(\tau) \quad (2.12)$$

and the power spectral density is determined as

$$S_{ww}(\omega) = \int_{-\infty}^{\infty} Q_w \delta(\tau) e^{-j\omega\tau} d\tau = Q_w \quad (2.13)$$

Equating (2.11) and (2.13) yields

$$Q_w = 2 \sum_{k=0}^{K} Q_{\beta_k} \left(\frac{T^{2k-1}}{2k+1}\right) \left[\frac{T^2}{2} - \frac{T^4 \omega^2}{24} + \frac{T^6 \omega^4}{720} - \cdots\right] \quad (2.14)$$

from which, by setting $\omega = 0$, one may form the approximate relationship

$$Q_w = \sum_{k=0}^{K} Q_{\beta_k} \left(\frac{T^{2k+1}}{2k+1}\right) \quad (2.15)$$

This is one of the new relationships developed to possibly yield a more accurate discrete representation of continuous white noise processes. Figure 2 shows the representation of the continuous and discrete white noise processes, including sample functions, autocorrelation functions, and the power spectral densities.

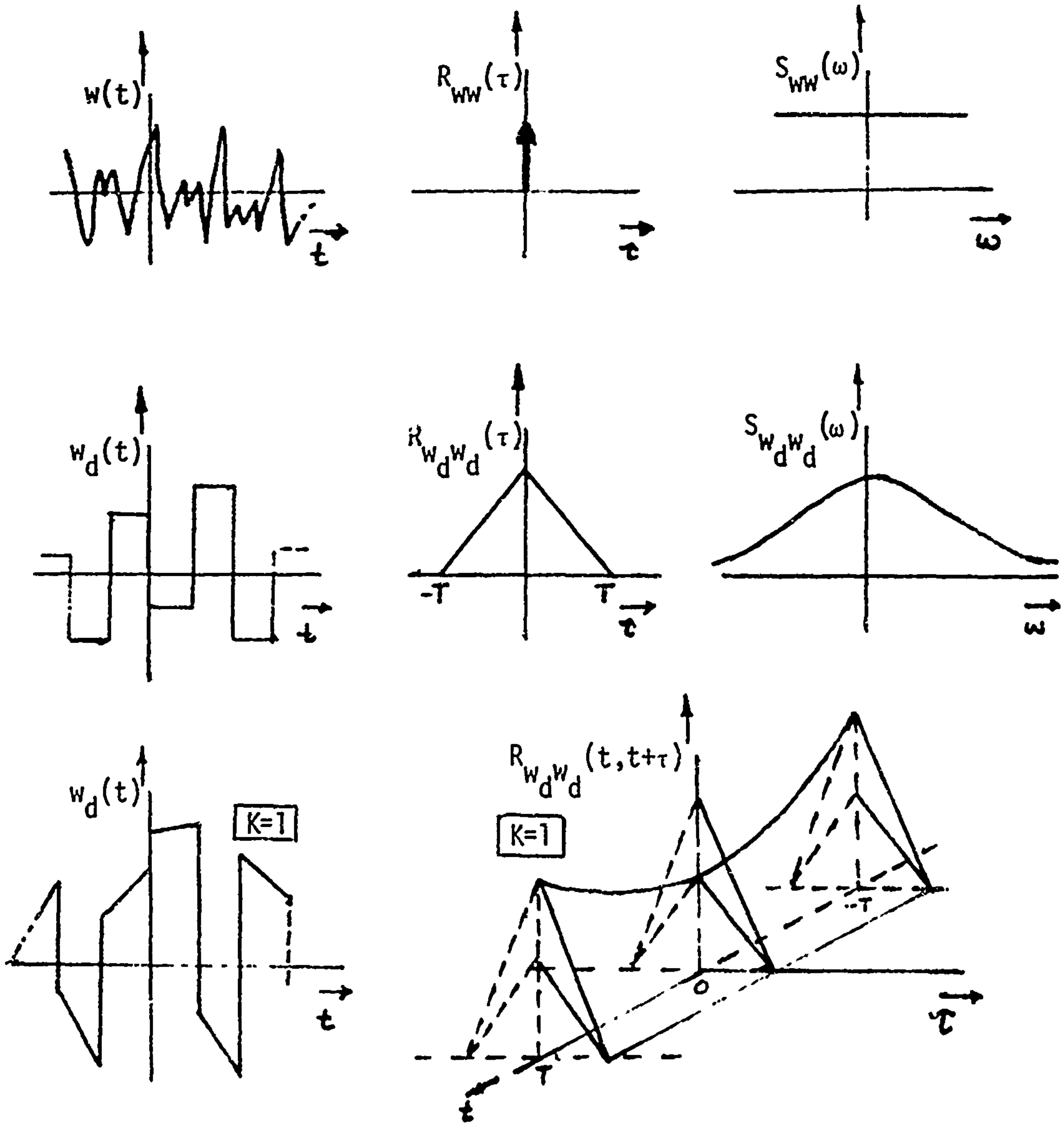


Figure 2. Continuous and Discrete White Noise Representations

Another method was developed towards the improvement of the discrete representation of continuous white noise processes. Consider the random process y(t) given by

$$y(t) = A \cos(\alpha t + \theta) \tag{2.16}$$

where A is a Gaussian random variable with variance σ_A^2 and a mean of zero, α is a constant, and θ is uniformly distributed on the range $(0, 2\pi)$. A and θ are assumed to be independent. It can easily be shown that

$$R_{yy}(\tau) = \begin{cases} \frac{\sigma_A^2}{2} \left(1 - \frac{|\tau|}{T}\right) \cos(\alpha\tau) & \text{for } |\tau|<T \\ 0 & \text{Otherwise} \end{cases} \tag{2.17}$$

Suppose a discrete random sequence $w_d(t)$ is generated by applying (2.16) on an interval-by-interval basis. This sequence may be used to approximate a given continuous white noise process as before by setting

$$\begin{aligned} Q_W &= 2 \int_0^T \frac{\sigma_A^2}{2} \cos(\alpha\tau) \cdot \left[1 - \frac{|\tau|}{T}\right] d\tau \\ &= \sigma_A^2 \left[\frac{1 - \cos(\alpha T)}{T\alpha^2}\right] \end{aligned} \tag{2.18}$$

This is the relationship developed for determining the variance of the discrete model. The simulation results of this method and the method developed earlier in the section are compared with the numerical results obtained earlier in (24). The method in (2.8) is referred to as the standard method, and the method developed in (2.9)-(2.15) is called the slope method. Furthermore, the alternate method in (2.16)-(2.18) is referred to as the cosine method.

Numerical Results

Consider the second-order, linear, time-invariant system described by

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -2x_1 - 3x_2 + w(t)\end{aligned} \qquad (2.19)$$

Recursive relationships used to generate the random input sequence w_d for the above second-order system have the form

$$Y_{i+1} = GY_i \qquad \text{(Modulo } M) \qquad (2.20)$$

Brown and Rowland (28) obtained satisfactory statistical properties from the pseudo-random number generator with $G = 19971$, $M = 2^{20}$, and $Y_o = 31571$. The generated numbers are uniformly distributed on (0,1). These numbers may be converted into a zero-mean, unity-variance Gaussian distribution by the exact closed-form relation developed by Box and Muller (6)

$$\begin{aligned}Z_1 &= (-2 \log_e Y_1)^{1/2} \cos 2\pi Y_2 \\ Z_2 &= (-2 \log_e Y_1)^{1/2} \sin 2\pi Y_2\end{aligned} \qquad (2.21)$$

where Y_1 and Y_2 are uniformly distributed, and Z_1 and Z_2 are Gaussianly distributed random variables.

Numerical results for this example are shown in Figure 3 with the average per cent error on the output variance ($\sigma_{x_1}^2$) versus the number of Monte Carlo runs for the three methods being compared. Using a step size T of 0.05, the standard method utilized pseudo-random numbers with a variance Q_{w_d} of Q_w/T equal to 20. The case of $K = 1$ was used for the slope method with the random variables

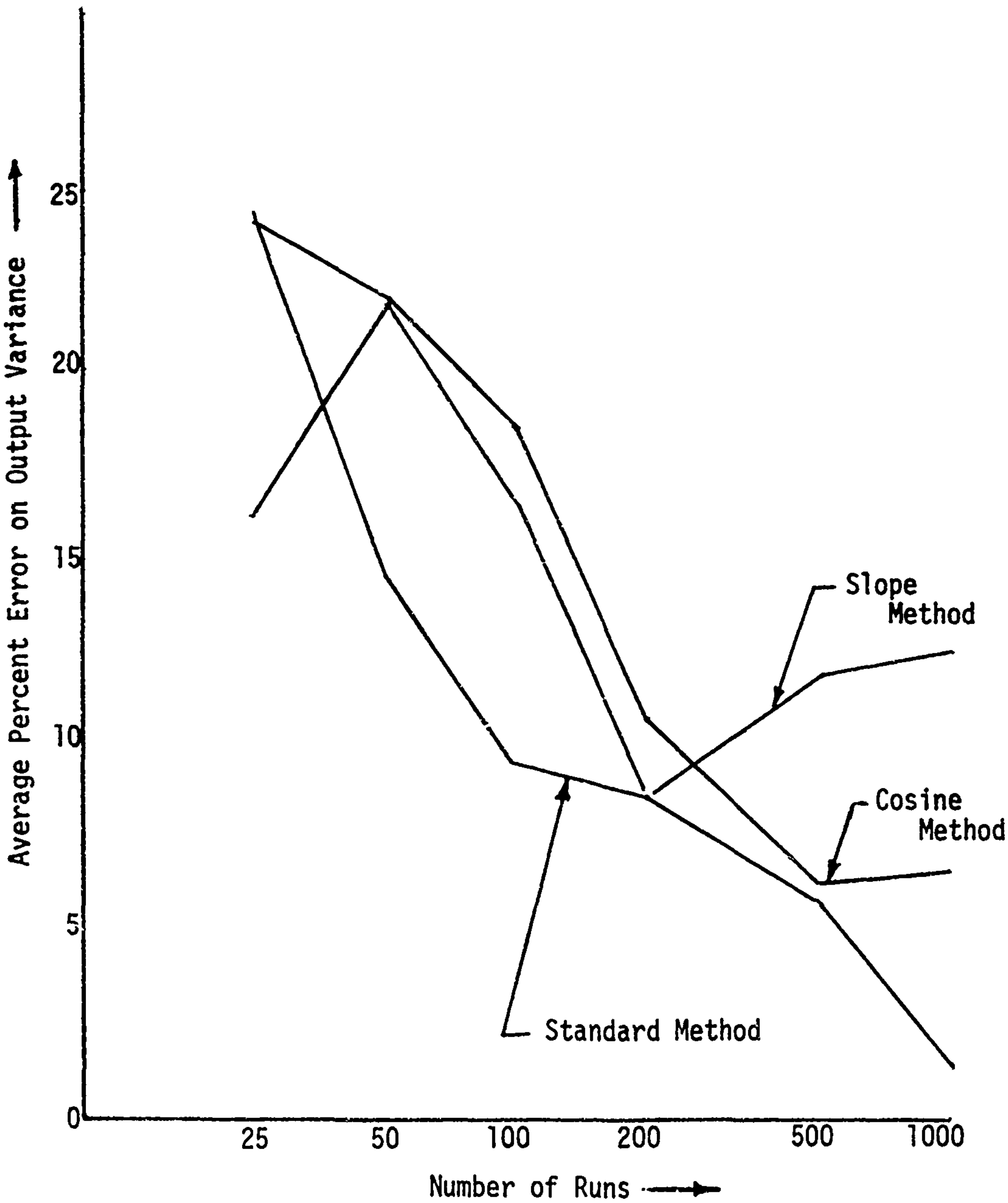


Figure 3. Average Percent Error on the Output Variance by the Monte Carlo Technique

β_0 and β_1 being given equal weight. Several other cases (K = 2,3, and 4) with several alternate weighting methods for the β's were also simulated, but no significant improvement was obtained. The results of the cosine method shown in Figure 3 used $\sigma_A^2 = 6.44$, $\alpha = 4\pi$, and $T = 0.05$. Different combinations of α and σ_A^2 were also used in other runs without improvement. Moreover, the use of Z_1 and Z_2 from (2.21) in consecutive intervals as opposed to using only Z_1, as shown in Figure 3, failed to yield any improvement. Finally, using alternate values of Z_1 and/or Z_2 did not improve the results shown. Therefore, the standard method was the best of those tested in terms of accuracy. In addition, the standard method requires only a single pseudo-random number per interval, which results in a particularly simple implementation as shown in Appendix B.

Summary

The direct covariance algorithm was extended in this chapter for application to linearized variational equations about the noise-free solution for nonlinear systems. Numerical results showed that the algorithm is applicable to those nonlinear systems with low input noise levels and mild nonlinearities. A generalization (29) was proposed for improving the discretization procedure for simulating continuous white noise processes on the digital computer. Extensive Monte Carlo testing on a second-order system indicated that the standard method developed earlier was both superior in accuracy and the most efficient for implementation purposes. This efficient discretization procedure forms the basis for the subsequent Monte Carlo validation of the computer software package developed in Chapter III.

CHAPTER III

IMPLEMENTATION OF THE DIRECT COVARIANCE ALGORITHM FOR LARGE-SCALE SYSTEMS

This chapter deals with the large-scale implementation of the direct covariance algorithm derived in the Chapter I and extended in Chapter II. A method for obtaining the exact solution for large-scale linear systems is presented, and the problems in implementing this solution for large-scale nonlinear systems are identified. The basic computer software package is developed with a particular emphasis on its application to large-scale missile systems and is applied to a thirty-third order math model of a six degree-of-freedom air defense missile system. Special problems encountered in the propagation of noise through the seeker subprogram of the missile are described in detail.

Exact Solutions for Large-Scale Linear Systems

The direct covariance algorithm derived in Chapter I is repeated here for convenience as

$$\dot{P}(t) = A(t)P(t) + P(t)A^T(t) + B(t)Q_{\underline{w}}(t)B^T(t) \tag{1.10}$$

In component form, (1.10) becomes

$$\begin{pmatrix} \dot{p}_{11} \cdots \dot{p}_{1n} \\ \vdots \quad \vdots \\ \dot{p}_{n1} \cdots \dot{p}_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} \cdots a_{1n} \\ \vdots \quad \vdots \\ a_{n1} \cdots a_{nn} \end{pmatrix} \begin{pmatrix} p_{11} \cdots p_{1n} \\ \vdots \quad \vdots \\ p_{1n} \cdots p_{nn} \end{pmatrix} + \begin{pmatrix} p_{11} \cdots p_{1n} \\ \vdots \quad \vdots \\ p_{1n} \cdots p_{nn} \end{pmatrix} \begin{pmatrix} a_{11} \cdots a_{n1} \\ \vdots \quad \vdots \\ a_{1n} \cdots a_{nn} \end{pmatrix}$$

$$+ \begin{pmatrix} b_{11} \cdots b_{1m} \\ \vdots \quad \vdots \\ b_{n1} \cdots b_{nm} \end{pmatrix} \begin{pmatrix} q_{11} \cdots q_{1m} \\ \vdots \quad \vdots \\ q_{m1} \cdots q_{mm} \end{pmatrix} \begin{pmatrix} b_{11} \cdots b_{n1} \\ \vdots \quad \vdots \\ b_{1m} \cdots b_{nm} \end{pmatrix} \tag{3.1}$$

Since P(t) is a symmetric matrix, i.e. $p_{ij} = p_{ji}$, the number of component differential equations in (3.1) is n(n+1)/2, where n is the system order.

Equation (3.1) can be solved exactly for constant A and B matrices. Rewriting (3.1) in the vector form yields

$$\underline{\dot{p}}(t) = A' \, \underline{p}(t) + \underline{r} \tag{3.2}$$

where

$$\underline{p}(t) = \begin{pmatrix} p_{11}(t) \\ p_{12}(t) \\ \vdots \\ p_{nn}(t) \end{pmatrix}$$

and A' and $\underline{r}$ are functions of the components of A, B, and $\underline{Q}_w$ in (3.1). The solution of the linear vector differential equation in (3.2) may be written as

$$\underline{p}(t) = e^{A'(t-t_0)}\underline{p}(t_0) + \int_{t_0}^{t} e^{A'(t-\tau)}\underline{r}\, d\tau \tag{3.3}$$

where $e^{A'(t-t_0)}$ is the state transition matrix associated with $\underline{p}(t)$ in (3.2). This matrix exponential, sometimes denoted by $\Phi(t-t_0)$, may be evaluated as

$$e^{A'(t-t_0)} = I + A'(t-t_0) + \tfrac{1}{2}A'^2(t-t_0)^2 + \ldots \tag{3.4}$$

Example

Equation (2.19) may be expressed in vector-matrix form by identifying

$$A = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \quad ; \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad ; \quad Q_{\underline{w}} = (1)$$

Therefore, (3.1) becomes

$$\begin{pmatrix} \dot{p}_{11} & \dot{p}_{12} \\ \dot{p}_{12} & \dot{p}_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix} + \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix} \begin{pmatrix} 0 & -2 \\ 1 & -3 \end{pmatrix}$$

$$+ \begin{pmatrix} 0 \\ 1 \end{pmatrix} (1) \quad (0 \quad 1) \tag{3.5}$$

Corresponding to (3.2), (3.5) may be written as

$$\begin{pmatrix} \dot{p}_{11} \\ \dot{p}_{12} \\ \dot{p}_{22} \end{pmatrix} = \begin{pmatrix} 0 & 2 & 0 \\ -2 & -3 & 1 \\ 0 & -4 & -6 \end{pmatrix} \begin{pmatrix} p_{11} \\ p_{12} \\ p_{22} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{3.6}$$

Using (3.3), the solution to (3.6) for P(0) = 0 is

$$\underline{p}(t) = \begin{pmatrix} \frac{1}{12} - \frac{1}{2} e^{-2t} + \frac{2}{3} e^{-3t} - \frac{1}{4} e^{-4t} \\ \frac{1}{2} e^{-2t} - e^{-3t} + \frac{1}{2} e^{-4t} \\ \frac{1}{6} - \frac{1}{2} e^{-2t} + \frac{4}{3} e^{-3t} - e^{-4t} \end{pmatrix} \tag{3.7}$$

Note that $e^{A'(t-t_0)}$ has $n^2(n+1)^2/4$ elements for an nth order system, which expands the computer storage requirements considerably beyond that required by using the matrix equation in (1.10) to solve for P(t) by numerical integration. For example, if n = 33, then P(t) may be obtained from (1.10) by solving 561 equations, whereas

$e^{A(t-t_0)}$ would require in excess of one-quarter of a million state transition matrix element evaluations. Moreover, if A and B are not constant in time, then the determination of the exact solution of P(t) in (3.2) is generally not possible. Since some components of A(t) and B(t) are always functions of time for nonlinear systems, the use of a suitable numerical integration formula, such as the fourth-order Runge-Kutta algorithm, is recommended for determining P(t) from (1.10) in general nonlinear cases.

The Basic Software Package

The considerations that were made during the development of the software package included obtaining accurate results while using a minimum amount of computer time, satisfying equipment requirements, such as computer storage, and determining the range of applicability for the direct algorithm on nonlinear systems.

The covariance matrix equation (1.10) was integrated along the nominal trajectory by using an integration step size for the covariance equations initially as half that of the system equations. The coefficient matrix A(t) for the system equations is a sparse matrix in many applications. For any large-scale system the coefficient matrix elements may be categorized as either zero, non-zero constants, non-linear functions of the nominal states, or implicitly related to the nominal states. For example, the thirty-third order missile system considered here had 920 zero coefficient matrix elements, which were neglected during program computations. In addition, constant elements were defined in the beginning of the program and left unchanged thereafter. The coefficient matrix was computed at each integration interval along with the nominal solution to yield a considerable savings in computer storage over the method of storing the A(t) matrix for all time t. Thus, each nonlinear element of A(t) was updated during each interval. Finally, those coefficient matrix elements which are related to certain state variables only implicitly, i.e. the functional relationship is available only via complicated computer programmed statements, were computed numerically at each interval. Additional details will be provided following the description of the large-scale application in the next section.

The application of the direct covariance algorithm to the thirty-third order nonlinear missile system yielded only approximate results because the accuracy of the direct covariance algorithm for nonlinear systems depends entirely upon the relative accuracy of the linearizing approximation for incremental variations about the noise-free solution. The error in the direct covariance results increases as the nonlinear terms in the exact incremental equation become more significant. The time-varying coefficient matrix prohibits the use of the state transition matrix equations. Thus, an accurate numerical integration technique was needed to integrate the n(n+1)/2 equations for the symmetrical covariance matrix.

The basic approach in the development of the software package is shown in Figure 4 in the form of a flow chart. The Fortran listing of this computer software package applied to a thirty-third order math model of a six degree-of-freedom air defense missile system is given in Appendix C.

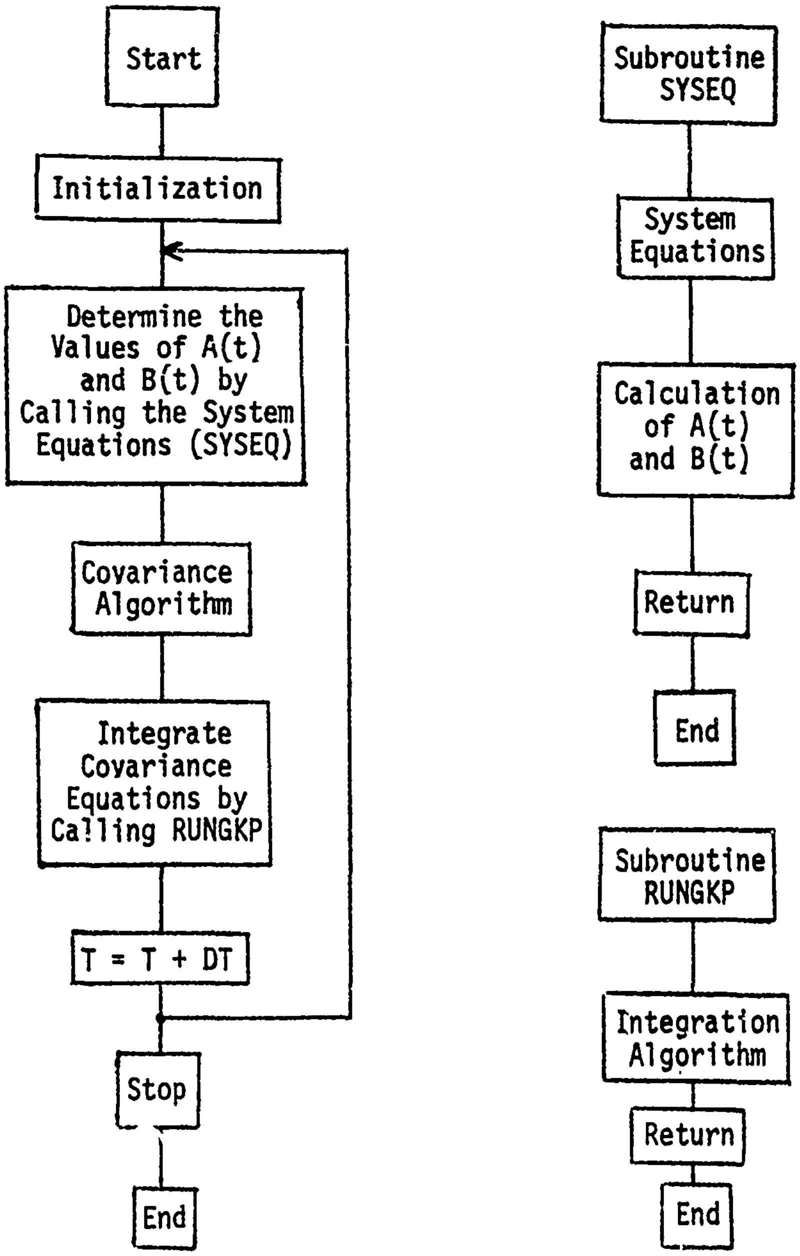


Figure 4. Flow Chart for the Development of the Computer Software Package

Description of the Missile System Application

The large-scale system investigated here is a thirty-third order math model of a six degree-of-freedom air defense missile system. The autopilot subprogram is fifteenth-order, the airframe subprogram which includes the missile rotational variables, the translational equations of motion, and launcher dynamics is twelfth-order, the seeker is second-order, and the actuator subprogram is fourth-order. The block diagram for the thirty-third order missile system is shown in Figure 5 with details of the autopilot and actuator in Figure 6. The target routine shown in the figure calculates the target-to-missile relative position and speed and generates line-of-sight signals.

Table I identifies all states of the missile system and assigns a specific number to each state. For example, the missile altitude Z is defined as the twenty-first state and occurs in the airframe subprogram. Table II provides the complete categorization of all elements of A(t) as either zero, indicated by blank entries, constant values (C), nonlinear functions of the nominal trajectory (NL), or numerically computed (NC). The number and per cent contained in each category are summarized in Table III.

Computations for Implicitly Related Elements

Only those elements of the A(t) coefficient matrix which are implicitly related to certain variables are computed numerically. For the thirty-third order math model of the six degree-of-freedom air defense missile system, the numerically computed elements are denoted in Table III by NC. The state identification of these state variables

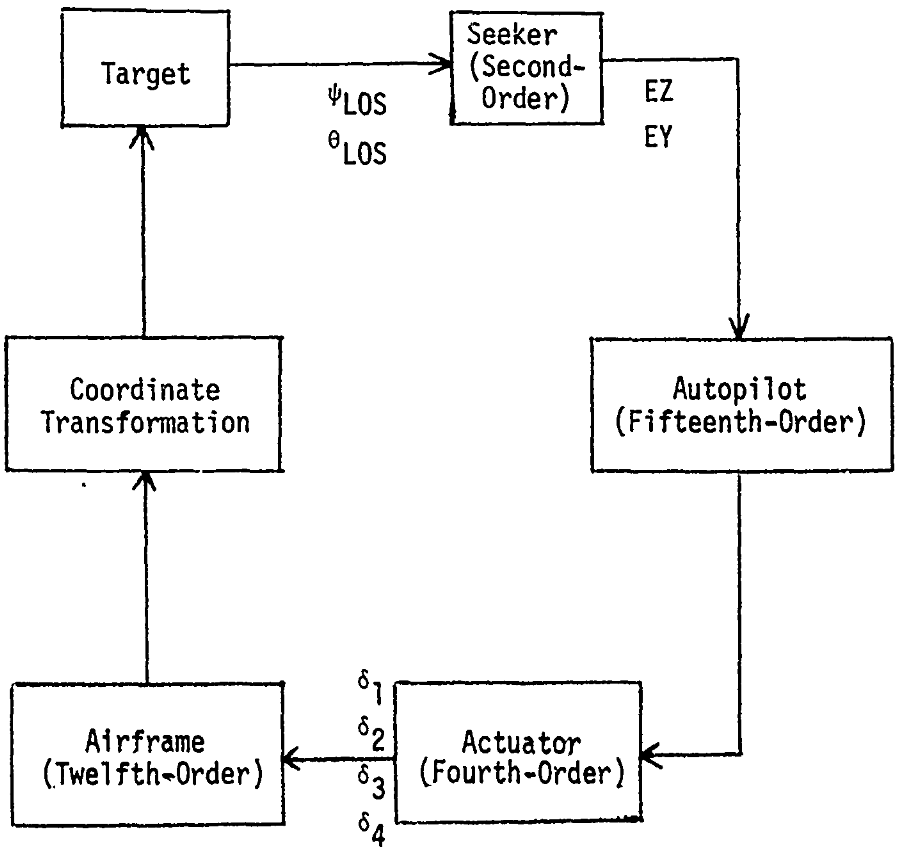


Figure 5. Block Diagram for the Thirty-Third Order Missile System

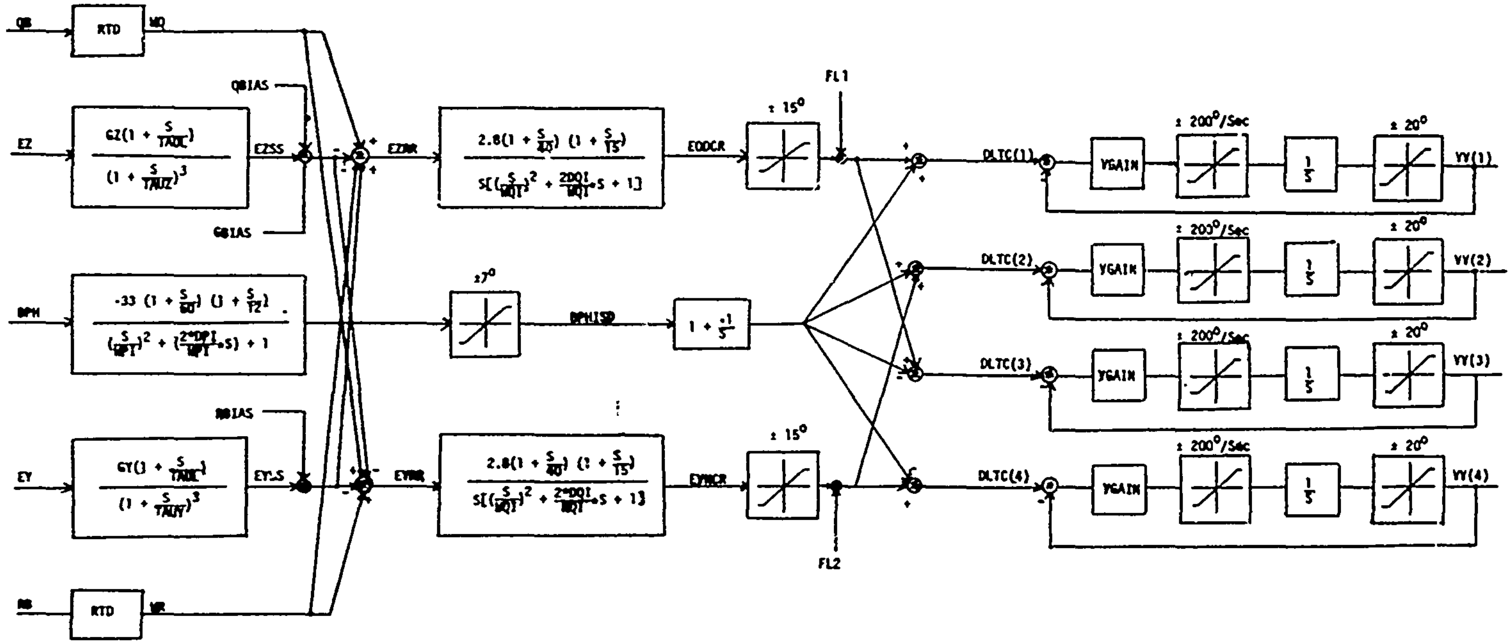


Figure 6. Block Diagram for the Autopilot and Actuators

TABLE I

DEFINITION OF THE MISSILE SYSTEM STATE VARIABLES

Subprogram	Description of State Variables	State Identification Name	State Identification
I. Autopilot	Guidance Pitch Filter	ZP1	1
		ZP2	2
		ZP3	3
	Guidance Yaw Filter	ZY1	4
		ZY2	5
		ZY3	6
	Roll Compensation	ZR1	7
		ZR2	8
		BPHIS	9
	Pitch Integrator	ZPI1	10
		ZPI2	11
		EODCR	12
	Yaw Integrator	ZYI1	13
		ZYI2	14
		EVNCR	15
II. Airframe	State Variables for Evaluating the Translational Equations of Missile Motion.	UE	16
		VE	17
		WE	18
		X	19
		Y	20
		Z	21
	Missile Rotational Variables	PB	22
		QB	23
		RB	24
	Euler Angles	THETA	25
		PHI	26
		PSI	27
III. Actuator	Vane Module Variables	VV(1)	28
		VV(2)	29
		VV(3)	30
		VV(4)	31
IV. Seeker	Internal States	VS(1)	32
		VS(2)	33

TABLE II

COEFFICIENT MATRIX FOR THE MISSILE SYSTEM

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	C	C	C																														
2	C																																
3		C																															
4				C	C	C																											
5				C																													
6					C																												
7							C	C																		C							
8							C																										
9							NL	NL																		NL							
10		C	C		C	C				C	C												C	C									
11										C																							
12		C	C		C	C				C	C												C	C									
13		C	C		C	C							C	C									C	C									
14													C																				
15		C	C		C	C							C	C									C	C									
16																NC	NC	NC			NC				NC	NC	NC	NC	NC	NC	NC		
17																NC	NC	NC			NC	NC*	NC*	NC*	NC	NC	NC	NC	NC	NC	NC		
18																NC	NC	NC			NC	NC*	NC*	NC*	NC	NC	NC	NC	NC	NC	NC		
19																C																	
20																	C																
21																		C															
22																NC	NC	NC			NC	NC			NC	NC	NC	NC	NC	NC	NC		
23																NC	NC	NC			NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC		
24																NC	NC	NC			NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC		
25																							NL	NL		NL							
26																						C	NL	NL	NL	NL							
27																							NL	NL	NL	NL							
28							NL	NL	NL			NL														NL		NL					
29							NL	NL	NL						NL											NL			NL				
30							NL	NL	NL			NL														NL				NL			
31							NL	NL	NL						NL											NL					NL		
32																																	
33																																	

TABLE III

CATEGORIZATION OF COEFFICIENT MATRIX ELEMENTS

Categorization	Number	Percentage
Zero Elements	920	84.5%
Constant Elements	52	4.8%
Nonlinear Elements	38	3.5%
Implicitly Related Elements	79	7.2%
Total	1089	100.0%

is given in Table I. The elements labelled NC* in Table III are computed to modify the derivatives when launcher dynamics of the missile system are in effect and are equated to zero after the second lug leaves the launcher. Numerically, the partial derivatives for A(t) in (2.6) are given by

$$A(t) = \frac{\underline{f}(\underline{x}_N + \underline{\Delta x}, \underline{\eta}_W, t) - \underline{f}(\underline{x}_N, \underline{\eta}_W, t)}{\underline{\Delta x}} \tag{3.8}$$

where the notation $\underline{\Delta x}$ represents small perturbations about the nominal flight path $\underline{x}_N(t)$. These perturbations have small lower limits when P(t) is very near zero, but $\underline{\Delta x}$ is increased by adding one-tenth of the standard deviation of the particular state under consideration when P(t) is set near zero. Therefore, the numerically computed elements of A(t) result in an adaptive feature for the direct covariance algorithm.

The large number of sequential calculations for the noise propagation equations results in numerical problems which can be handled most effectively by using double-precision throughout. To avoid these time consuming operations, the elements in a particular column of P(t) were arbitrarily set to zero whenever the corresponding diagonal element was below 10^{-10}.

Seeker Noise Considerations

For the noise propagation studies, the noise was introduced at four places in the missile system. The first two places are shown in Figure 6, and the other two white noise inputs were added to the seeker subprogram of the missile system. These latter two noise inputs involved perturbing the line-of-sight signals ψ_{LOS} (BEPSZ) and θ_{LOS}(BEPSY) generated by the target subprogram as shown in Figure 5.

These noise signals were passed through the dead-zone as shown in Figure 7. Two subprograms which were developed to obtain the variance of noise after passing it through the dead-zone are included in Appendix C as Subroutines SNOISE and DETARA. These subprograms utilize the three cases depicted in Figure 8 in which the nominal values of BEPSZ or BEPSY lie below -TMP1, between -TMP1 and +TMP1, or above +TMP1. The density functions of EZ and EY are each composed of three impulses at SKSP or SKSY, zero, and -SKSP or -SKSY. The weighting on each of these impulses is determined by the area of the Gaussian input signals lying within the different ranges of the dead-zone nonlinearity as shown in Figures 7 and 8. The calculation of this area is performed in Subroutine DETARA. It should be emphasized that the dead-zone is a very harsh nonlinearity, which can result in a severe test in applying the direct covariance algorithm. However, the seeker noise was injected at this point in the system because such noise disturbances do occur in the actual missile system.

The operation of Subroutines SNOISE and DETARA is described here to demonstrate how to handle noise propagation across a dead-band relay element. Card 16 defines SIGBEP in terms of the seeker noise input standard deviation (VNOISD), SGTMP1, and VBEPS. The latter two terms are standard deviations of the noises due to the random effects of the position coordinates X, Y, and Z and the seeker state variables, respectively. Cards 10 through 15 yield the expression for SGTMP1 in terms of the covariance matrix associated with the X, Y, and Z states. It has been assumed that these states are Gaussianly distributed and, therefore, that their fourth central moments are equal to three times their respective variances. Similarly, the contribution of

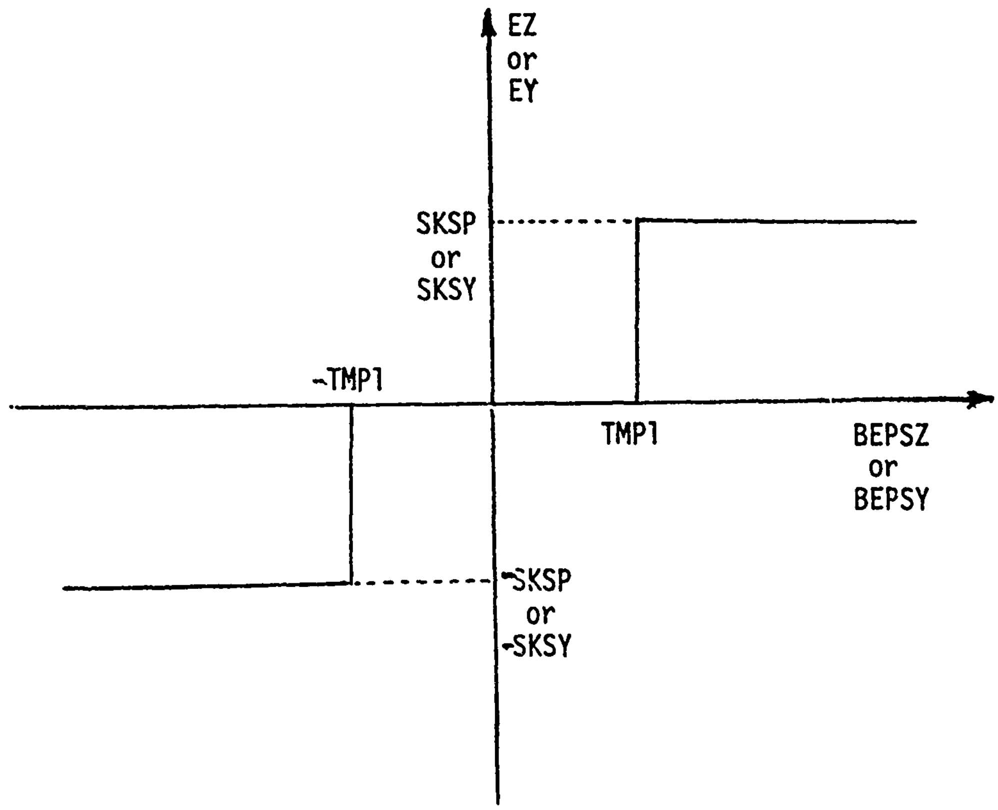


Figure 7. Dead-Zone Details Used in the SEEKER Subprogram

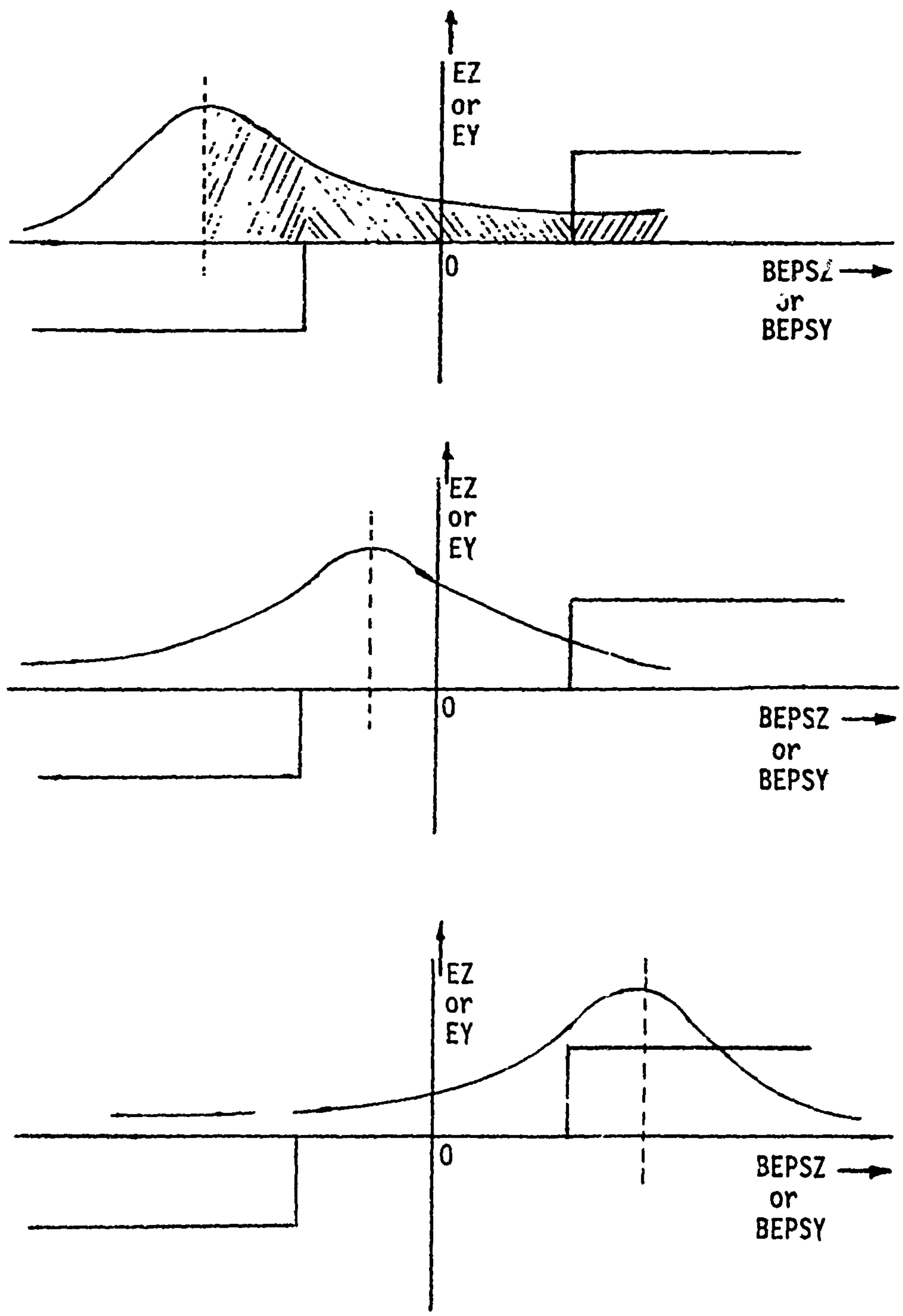


Figure 8. The Effects of the Dead-Zone Nonlinearity on Seeker Noise Inputs

the noisy seeker states are handled by using the same procedure. Cards 17 and 27 identify the region of operation of the seeker relays. For example, if the relay output EC equals -SKSP, then the relay is switched to the negative region as shown in the upper diagram of Figure 8. The distance DIST between BEPS, which represents the mean of either BEPSZ or BEPSY depending on which of the two seeker relay nonlinearities is being considered, is defined in Card 18 as DIST = -TMP1 - BEPS. Normalizing this Gaussian density curve by dividing by the standard deviation SIGBEP to give POS, one may use standardized Gaussian density tables to determine the area under the curve below -TMP1, the area between -TMP1 and +TMP1, and the area above +TMP1. Specifically, Card 20 yields the desired area (AL1) from Subroutine DETARA. Note that the total probability of BEPS lying below -TMP1 is one-half plus that area just determined from DETARA (Card 21). Card 22 defines POS for the curve between the actual BEPS and +TMP1. The resulting area (A01) is the sum of AL1 determined above and the desired dead-band area A0. Therefore, A0 = A01 - AL1 as given by Card 24. Moreover, since the sum of the total area under the curve is unity, the probability that BEPS lies above +TMP1 is AU = 1-AL - A0 (Card 25). Similarly, the probabilities associated with the other cases shown in Figure 8 may be calculated. Finally, Cards 45 through 47 compute the mean of EC (SIGEC), the second moment of EC (SGSEC), and the variance of EC (SGSQ).

Summary

The general framework for implementing the direct covariance algorithm for large-scale systems has been described in this chapter. Numerical results to be presented later have indicated that the two seeker nonlinearities are of major importance in determining nonlinear operating characteristics of the thirty-third order missile system. In particular, for seeker input noise variances of (2 degrees)^2, the direct covariance algorithm is applicable to a large range of operations during the mid-portion of a typical flight. Chapter IV describes the details of a digital computer software package which combines Monte Carlo runs for the first and last parts of the flight with direct covariance algorithm results for the mid-portion of the flight. Numerical results from this combined program are then presented in Chapter V.

CHAPTER IV

COMBINED MONTE CARLO - DIRECT COVARIANCE ALGORITHM SOFTWARE PACKAGE DESCRIPTION

The digital computer software package is described in this chapter initially in terms of a computer flow chart of the complete program. The general effects of incorporating the direct covariance algorithm into an existing digital computer program are identified, and subroutines are grouped according to whether major or minor changes are needed to realize the combined algorithm. Finally, details of these changes are provided, and a description of the resulting control cards is given for a variety of simulation run conditions.

Computer Flow Chart

General computer software operations are described in Figures 9 through 12. The basic diagram for all operations is shown in Figure 9, while nominal flight conditions, Monte Carlo simulations, and covariance calculations are given in Figures 10, 11, and 12, respectively. The combination simulation run indicated as a fourth option in Figure 9 is obtained by using appropriate control cards which combine the operations described in Figures 11 and 12 over designated portions of the flight.

Subroutine Descriptions

The types of changes needed to convert a digital computer program which yields only nominal trajectory information, i.e. without noise, are indicated in Table IV. Descriptions of these changes in individual subroutines are provided in the following paragraphs.

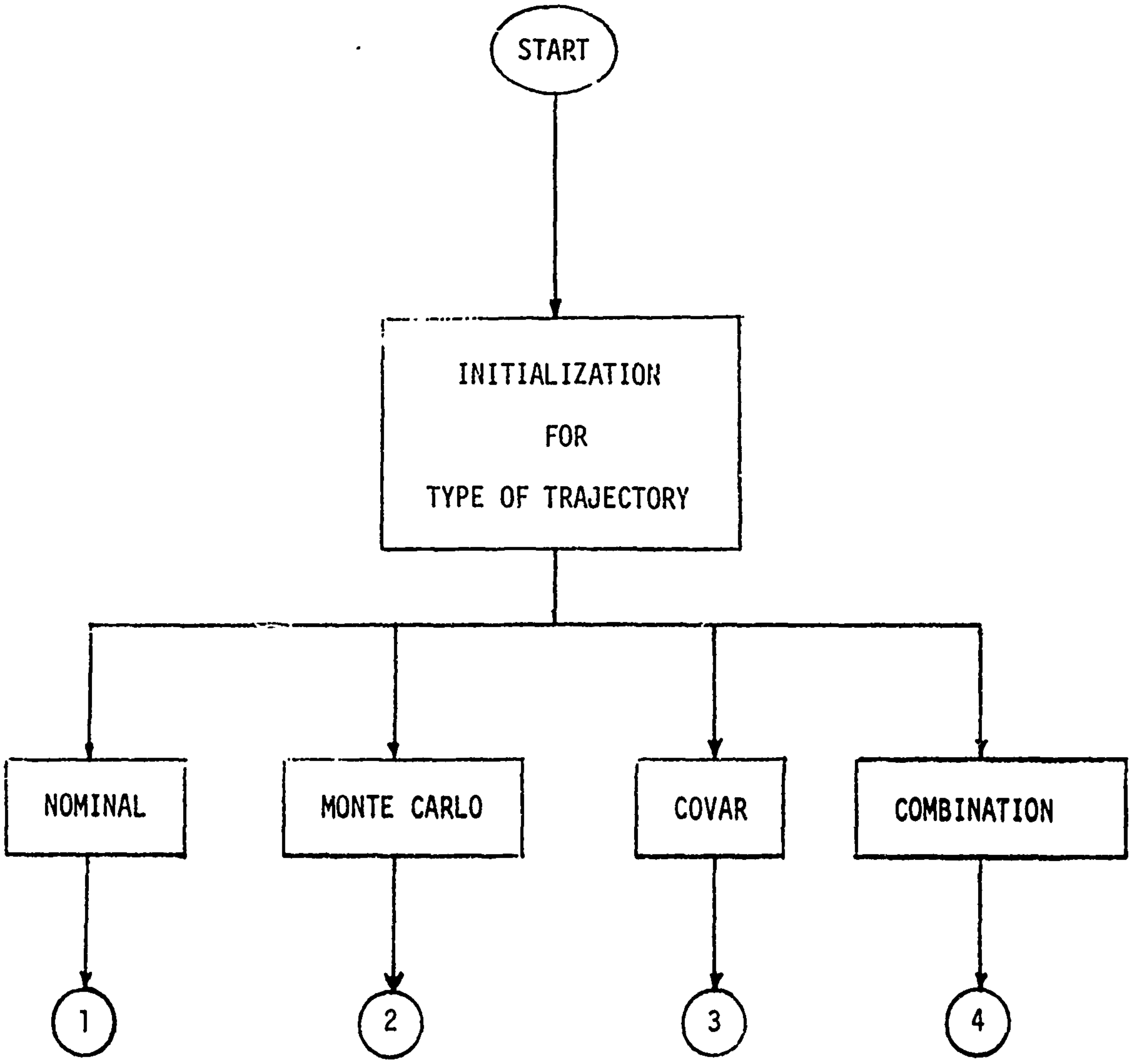


Figure 9. Flow Chart for General Computer Software Package

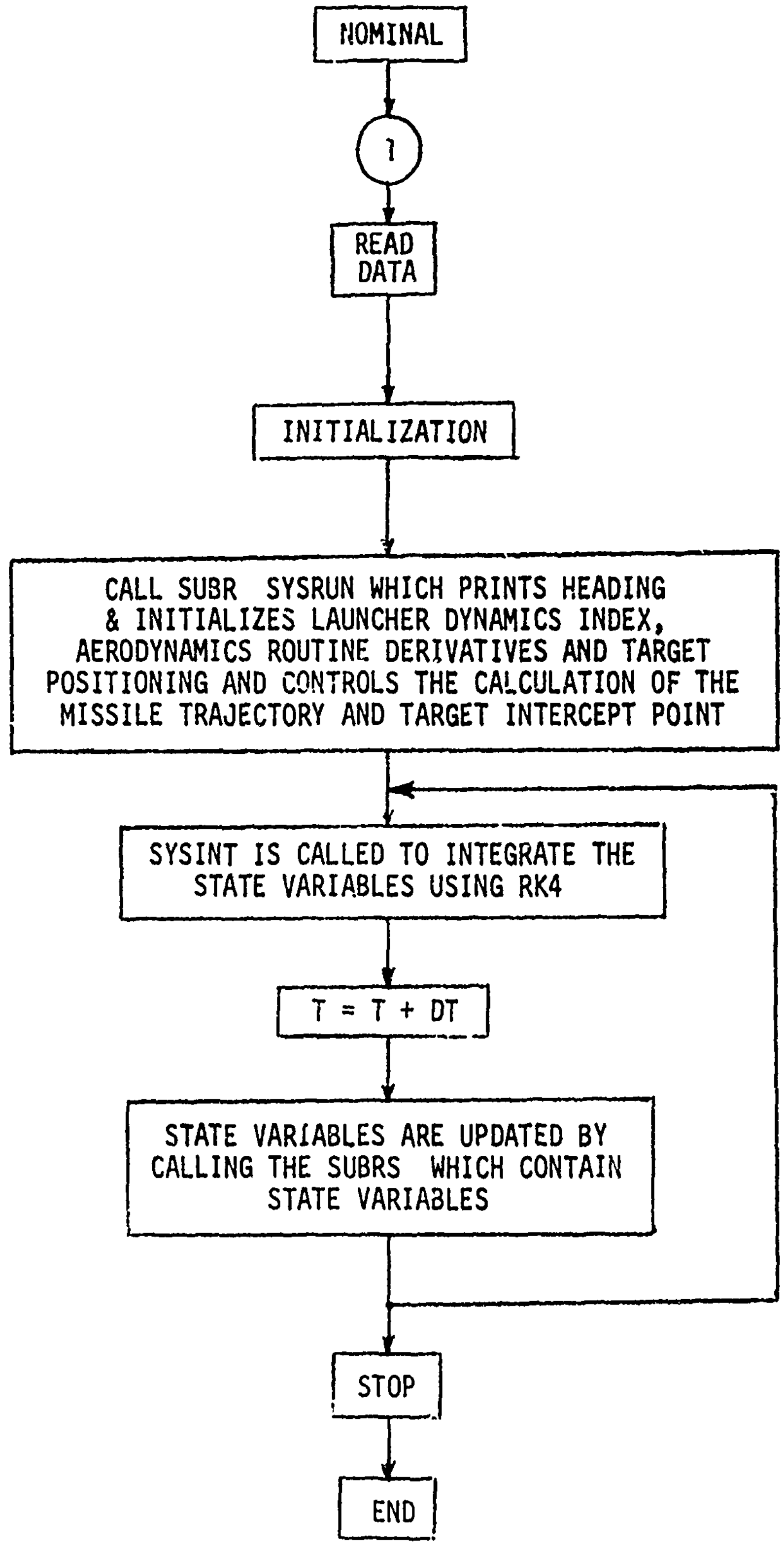


Figure 10. Flow Chart for Nominal Flight

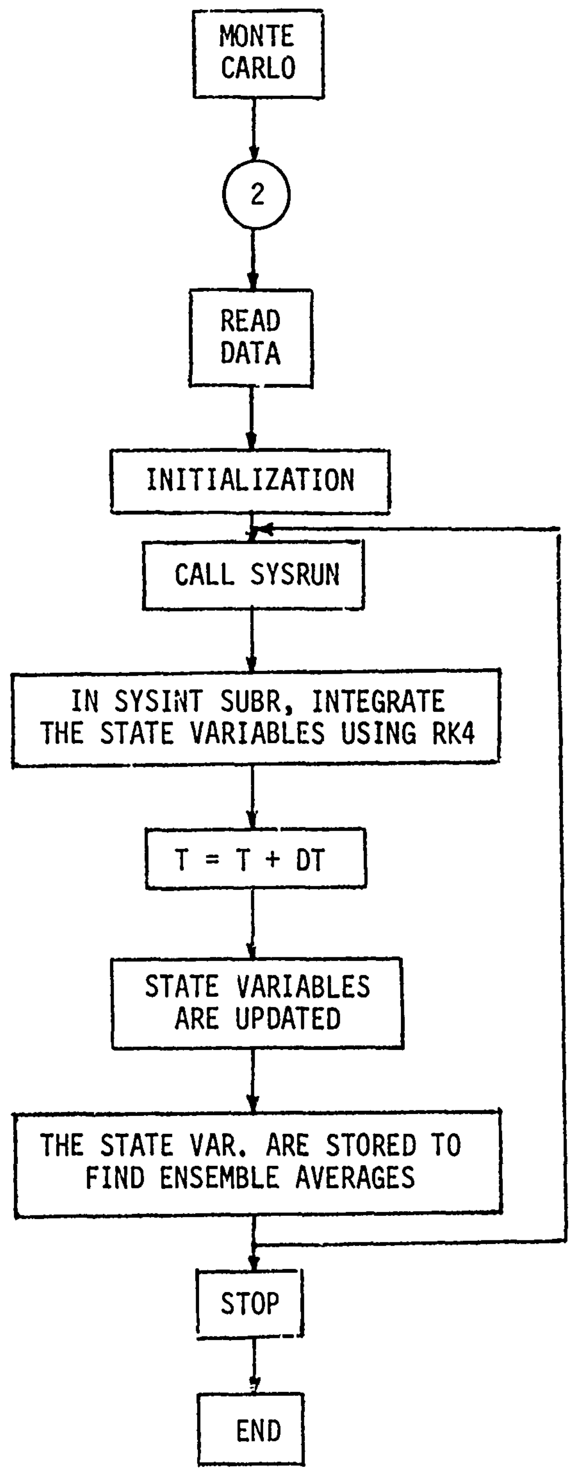


Figure 11. Flow Chart for Monte Carlo Simulations

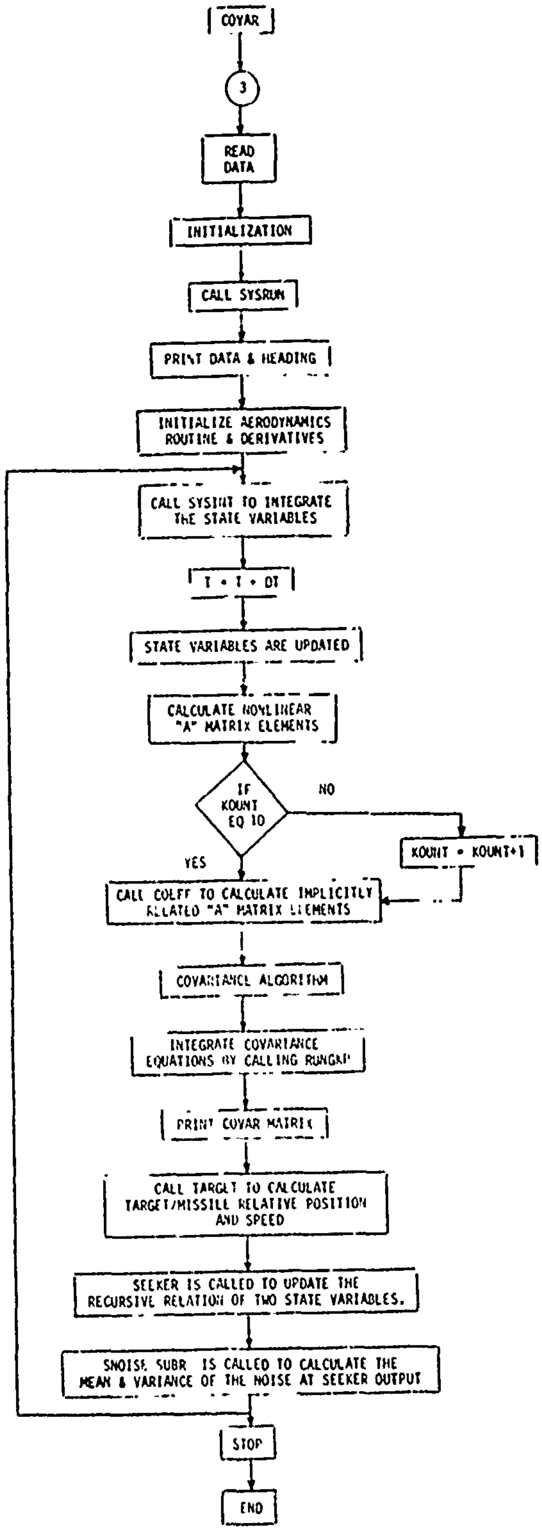


Figure 12. Flow Chart for Direct Covariance Algorithm.

TABLE IV

SUBROUTINE CLASSIFICATION

MAJOR CHANGES	MINOR CHANGES	NEW SUBROUTINE	NO CHANGE
MAIN	SYSRUN	INTA2M	AUTOPT
TARGET	SEEKER	SNOISE	PRDATA
SYSINT	VANEMD	RANDU	ROTATM
	TRANSM	RANDG	DTLUX1
	AERODY	RUNGKP	FUNCTION DEAD
	BLOCK DATA	COEFF	FUNCTION XLIMIT
	THRCON	COVAR	
		DETARA	TRANS
		MDERIV	RK4
			INITIA
			INTRP3

MAIN

Subroutine MAIN takes care of all the initializations of the variables used during the flight. The run could be made as Nominal, Covariance, Monte Carlo or their combinations with proper initializations given by Cards 149-168 and 180-203. Cards 171-177 are initialized depending upon the type of run chosen. Cards 136-146 are used to read the initial values of the variables from the cards and to write them on the disc to be used later in the program for re-initialization during Monte Carlo runs. In Cards 206-243, various variables are initialized which are used in the program. The Thrust and Aerodynamic Tables are read in Cards 247-266. The initialization Subroutine INITIA is called in Card 272. The initialization for Monte Carlo runs are made in Cards 274-302. NUM number of Monte Carlo runs are made in Cards 303-355. The (NUM+1)th (NUM = number of Monte Carlo runs) entry in the DO loop is for re-initializations of the variables. Cards 357-375 are used to calculate ensemble averages and for print-out. The off-diagonal multivariate samples are generated for Monte Carlo simulations from specified covariance matrix calculations in Cards 383-399 and 418-420. If VTIME2 is greater than or equal to TSTOP, then in Card 381 the program is diverted to Card 465. If Monte Carlo runs are made in the latter part of the trajectory, Cards 401-455 make (NUM-1) runs and Card 380 makes the first run resulting in a total of NUM number of runs. The ensemble-averaging and print-out is achieved by Cards 456-464.

TARGET

In this subroutine Cards 51-79 have been added to calculate the variances of BEPSZ and BEPSY and their effects are incorporated into Subroutine SNOISE (Card 16). The details of Subroutines SNOISE and

DETARA have been given already in Chapter III of this report. The values of the variances VBEPSZ and VBEPSY are transferred to Subroutine SNOISE through Subroutine SEEKER in Cards 9, 22, and 28. Cards 51 and 52 allow the calculation of variances only for the covariance program. Cards 54-69 are used to break up the long expression of VBEPSZ and VBEPSY in Cards 70-74 and 75-79, respectively.

SYSINT

This subroutine calls the integration subroutine (RK4) to integrate all the state variable differential equations over one time step. This routine also calculates the nonlinear "A" matrix elements for the covariance program. The calculation of implicitly related "A" matrix elements are calculated by calling Subroutine COEFF. The direct covariance algorithm is obtained by calling the COVAR subroutine and is integrated by calling RUNGKP, which uses the RK2 integration method. For the calculation of the state mean and variance by Monte Carlo runs, the values of the state variables and their square are stored at different points in time and the ensemble average is calculated in MAIN.

Cards 17 to 32 have been added to transfer the variables to other subroutines as explained in the previous paragraph. Cards 36 to 56 are used to store the values of state variables at time VTIME2 to make Monte Carlo runs. Also, these values which were stored at time VTIME2 are printed the first time through the program. Cards 59 to 67 are used to calculate four normally distributed random number with unity variance and zero mean for Monte Carlo runs. These numbers are used in the VANEMD and TARGET subroutines. Cards 86 to 220 are used to calculate the nonlinear "A" matrix elements only the first time through the program. Cards 230 - 259 are used to calculate and integrate the covariance matrix, to check for negative diagonal elements, and for print out. Cards 261-281 are used to store the state variables and their squares at different points in time for Monte Carlo runs. These values are stored whenever N1 equals K1 in Card 264. Cards 282-294 are used to store the value of state variables only at the switching time VTIME1, which is needed to calculate off-diagonal terms of the covariance matrix. Cards 295-296 are used to store the time at which the state variable values were accumulated to find the ensemble-average.

SYSRUN

Only a few changes have been made in this subroutine. In Card 26 the value of KIT is initialized to zero in MAIN and transferred by a common block in Card 21. This value is changed only in Subroutine SYSINT Card 40 when the program is switched from covariance to Monte Carlo to see that the aerodynamics routine, derivatives and target position, etc., are not initialized when Monte Carlo runs are made for T greater than VTIME2. Card 63 makes sure that the K is reinitialized to 1 because the program bypassed Card 53. Cards 120-123 are used to control the program for Monte Carlo runs. The value of KONTER is altered only in Card 122. Once it attains the value equal to NUM, then KONTER is not altered thereafter. Cards 146-151 are used to print out the covariance matrix at that instant in time.

SEEKER

In this subroutine Cards 8 - 10, 20-23, and 26-29 were added to insert noise into the seeker, and Subroutine SNOISE is called to calculate the mean and variance across the nonlinearity. These values are only calculated when the covariance program is in operation. Otherwise, these cards are bypassed.

VANMED

In this routine noise is added in the vane modules when the Monte Carlo program is run. In Cards 15-18, normally distributed random numbers are calculated in Subroutine SYSINT by calling RANDG.

TRANSM

Card 25 is used while calculating implicitly related "A" coefficient matrix elements in the COEFF subroutine. The value of KK1 is initialized in MAIN to 1 and is only altered in COEFF and then again replaced by 1 at the end of the COEFF Subroutine. In Card 65 when KK3 is not equal to zero the program returns to the calling subroutine. KK3 is initialized in MAIN to zero and is passed through the common block Card 18. The value of KK3 is modified only in the COEFF subroutine and is replaced by zero at the end of this subroutine.

AERODY

Only two cards were added to this routine: Cards 19 and 23. The value of KK5 is passed through the common block in Card 19. The value of KK5 is initialized to zero in MAIN. This value is only modified in the COEFF subroutine for the calculation of the implicitly related "A" coefficient matrix elements. The value of KK5 is replaced by zero at the end of the COEFF subroutine.

BLOCK DATA

Cards 9 and 10 were added to initialize the step size and the number of state variables denoted by H and MS, respectively, in Card 9. The step size H is not used at present in the program but MS is used at various places throughout the program mainly for DO loops.

THRCON

Card 13 is added in the routine to preserve the values of THRP and TIMP while making calculation for "A" matrix elements in Subroutine COEFF. These values are preserved in COEFF by transferring them to other variables and replacing them at the end of calculations.

INTA2M

This new subroutine initializes the constant elements of the "A" matrix only once in the MAIN program through Card 243.

RANDU

This program generates normally distributed random numbers with zero means and unity variances. The random numbers equal in number to the number of state variables are generated and passed through variable YNORM to MAIN by Card 417. These are used for Monte Carlo runs after time VTIME2 to give random normally-distributed starting conditions at that point in time.

RANDG

This program also generates normally distributed random numbers with zero means and unity variances. These numbers are transferred through variable XNORM when called in Subroutine SYSINT through Cards 64 and 67. These normally distributed numbers are used to insert noise in the vane modules and the seeker during Monte Carlo simulations at locations in VANEMD by Cards 15-18 and in TARGET by Cards 48 and 49.

RUNGKP

This subroutine is an integration routine and the RK2 method of integration is used to integrate n(n+1)/2 equations where n is the number of state variables. This routine is called in SYSINT (Card 236).

The value of DTH is transferred from SYSINT via Card 231.

COEFF

This subroutine calculates the implicitly related "A" matrix coefficients. In all, 79 elements are calculated in this routine. The values of the KK's are defined in Cards 38-42. These are used throughout the program to control the required calculation of the "A" matrix elements. The nominal trajectory is perturbed slightly (Cards 43-46) to calculate the effect of this perturbation and thus obtain the "A" matrix elements. Card 47 sends the routine to Card 69 to store and preserve the nominal trajectory variables so that those values can be replaced after the calculations. Card 116 then sends the program to Card 342 to calculate the effect of the perturbation. In Card 358, Subroutine MDERIV is called only if LAUNCH is one or two. The "A" matrix elements denoted by NC* in Table II on Page 31 are equated to zero after LAUNCH is greater than 2 only once in Cards 362-366. Since the value of KK4 is one, the program goes from Card 359 back to Card 183. In Cards 183-191, the next value of the state variable is perturbed and the program goes to Card 117, where the A matrix elements are calculated. Since KK3 was 7, the program goes to Card 138 to replace the values of those variables which were stored and preserved earlier. The program again goes to Card 69 from Card 182 to repeat the same procedure for the next state variable.

COVAR

In this program the covariance algorithm is implemented. Since the P matrix is symmetrical, the lower triangle of the P matrix is equated to the upper triangle in Cards 11-13. In Cards 14-94 the AP matrix is calculated. In Cards 95-97 the PA^T matrix is obtained. Cards 98-100 give the $AP + PA^T$ matrix. The BQB^T terms are added in Cards 101-111.

MDERIV

This subroutine has been added to modify the derivatives when the launcher dynamics are in effect. It is called in the COEFF subroutine (Card 358) during the calculations of the implicitly related "A" matrix elements. This program is a part of the ROTATM subroutine (Cards 49-72) with a change of variables.

Summary

The details of the combined computer software package have been presented in this chapter. Flow charts have been provided to describe the nominal flight, Monte Carlo simulations and the direct covariance algorithm. It should be pointed out that Cards 149-203 in MAIN describe the necessary modifications to run any of these cases, including the combination run. Numerical results using this software package are given in the following chapter.

CHAPTER V

NUMERICAL RESULTS

Both preliminary and final numerical results are presented in this chapter for the six degree-of-freedom air defense missile system described in Chapter III. Initially, tradeoff considerations and simplifying approximations are given. Direct covariance runs on the range from one to two seconds into the flight are then presented for modifications yielding from thirty-first up to fifty-first order missile systems. Core and speed requirements for these different systems are identified. It is shown that the initial and terminal portions of the flight are too nonlinear for the application of the direct covariance algorithm and, therefore, that Monte Carlo simulations must be utilized on these highly nonlinear segments. Final numerical results are presented for the entire flight by using the combined software package of Chapter IV.

Tradeoff Considerations

The considerations that must be made during tradeoff studies are closely related to the criteria for comparison purposes presented in Chapter I. Since the information provided and the extension possibilities are fixed by selecting the direct covariance approach, only the remaining criteria of accuracy, computational speed, computer storage, and program complexity may be used for tradeoff possibilities.

Accuracy

Accuracy plays a major role in achieving computational efficiency, since it has an inverse relationship with the computational speed. For example, trading accuracy for computational speed by changing the integration method from the fourth-order Runge-Kutta formula (RK4) to the second-order Runge-Kutta formula (RK2) may reduce the computation time considerably for large-scale systems. In any simulation problem the minimum acceptable accuracy level limits the maximum integration step size that may be chosen. Tradeoffs for the large-scale system are also influenced by the fact that direct covariance technique gives exact results for linear systems while the errors in the results of nonlinear systems depend on the amount of nonlinearity and the input noise level. In addition to the choice of integration method and the selection of the step size, the frequency at which the coefficient matrix is updated affects the algorithm accuracy.

Computational Speed

Tradeoffs may be used to minimize the computer time needed for the large-scale simulation and the application of the direct covariance algorithm. For the developed software package, the integration time needed for the covariance matrix equations may be reduced by nearly one-half by changing the integration method from RK4 to RK2, as mentioned earlier. A savings in computer time is also obtained by categorizing the coefficient matrix elements as zero, constants, nonlinear, and implicitly related to the state variables. Since the A(t) matrix is usually a sparse matrix, many coefficient elements are zero and thus neglecting them entirely during the calculations reduces the computer time considerably. Table III summarizes this categorization for the thirty-third order missile system described in Chapter III. Finally, further reductions in computational time may be achieved by calculating the A(t) coefficient matrix elements after every few integration intervals instead of every integration interval.

Computer Storage

The computer storage needed for applying the software package to the large-scale system can also be reduced by tradeoff. The general implementation of the direct covariance algorithm for large-scale systems requires a much higher computer storage as compared to a particular implementation. For an nth-order system, storing the large A(t) and B(t) matrices requires a large amount of computer storage. This may be reduced by deleting the zero elements and either converting these matrices into smaller matrices or to vector form. However, this procedure would tend to increase the complexity of the computer software package.

Program Complexity

The program complexity is another measure of an efficient computer software package. The general implementation of the direct covariance algorithm may reduce the program complexity to a minimum, whereas a particular implementation makes it quite complex. The complexity also increases, as noted above, by converting A(t) and B(t) in smaller matrices or vector form. Thus, a balance must be reached by trading accuracy, computational time, computer storage, and program complexity to provide a computationally efficient final software package.

Program Simplifications

Simplifying approximations were used for speeding up the direct covariance program. The use of constant coefficients in place of slowly-varying coefficients in the variational equations and neglecting extremely small coefficients entirely were approximations that were examined. In particular, 28 of the 38 nonlinear elements of the incremental coefficient matrix A(t) were held constant throughout the flight period of interest without a serious degradation in results. Furthermore, 18 of the 79 numerically computed elements were also simplified, and their effect was negligible on the performance of the direct covariance software package. Finally, the possibility of computing the "A" matrix elements at different varying intervals was investigated, but it was shown that the necessary overhead operations made such a procedure unfeasible.

The calculation of all "A" matrix elements automatically i.e. numerically, was shown to require a computation time that was much too long. However, such operations yield, in general, the simplest possible program. For a fifty-first order missile system, this simplest program for computing all 2601 "A" matrix elements requires approximately 27 minutes on the Sigma 5 Computer for computations in the range between 1 second and 1.1025 seconds into the flight. The minimum computational time possible was only approximately 5 minutes obtained by using constants and nonlinear expressions wherever possible as indicated by Table III in Chapter III. Also, the zero elements were not computed. The resulting program was obviously more complex than the general program. An intermediate possibility which required approximately six minutes for the given calculation was also identified by eliminating a large number of the zero-element calculations but including certain of these elements when they are grouped within a given block of non-zero elements. For the direct covariance algorithm, approximately 36K words of core (including the monitor) are needed to perform noise propagation calculations for systems up to fifty-first order.

Preliminary Numerical Results

Significant problems were encountered in implementing the direct covariance algorithm for the initial portion of the flight. These problems are discussed in detail later in this section. Because of these problems, comparisons between Monte Carlo simulations and covariance runs were made on the range between one and two seconds into the flight. Numerical results are shown in Figure 13 for several orders of missile systems.

The thirty-first order system was obtained from the thirty-third order system in Figure 5 by neglecting the dynamics of the second-order seeker subprogram. The thirty-seventh order system included the addition of two second-order filters (pitch and yaw rate gyros) in the autopilot. Tests were also made by using two seventh-order colored noise prefilters for the actuator noise inputs to yield a fifty-first order system. The comparisons between Monte Carlo simulations and these covariance results indicate that existing errors may be attributed to the use of only 25 Monte Carlo runs. These tests were made by using seeker input noise signals with variances of (2 degrees)^2, which are later shown to yield excessive miss-distances. The seeker characteristics used earlier in a terminal homing simulation on the hybrid computer at the U. S. Army Missile Command had noise variances

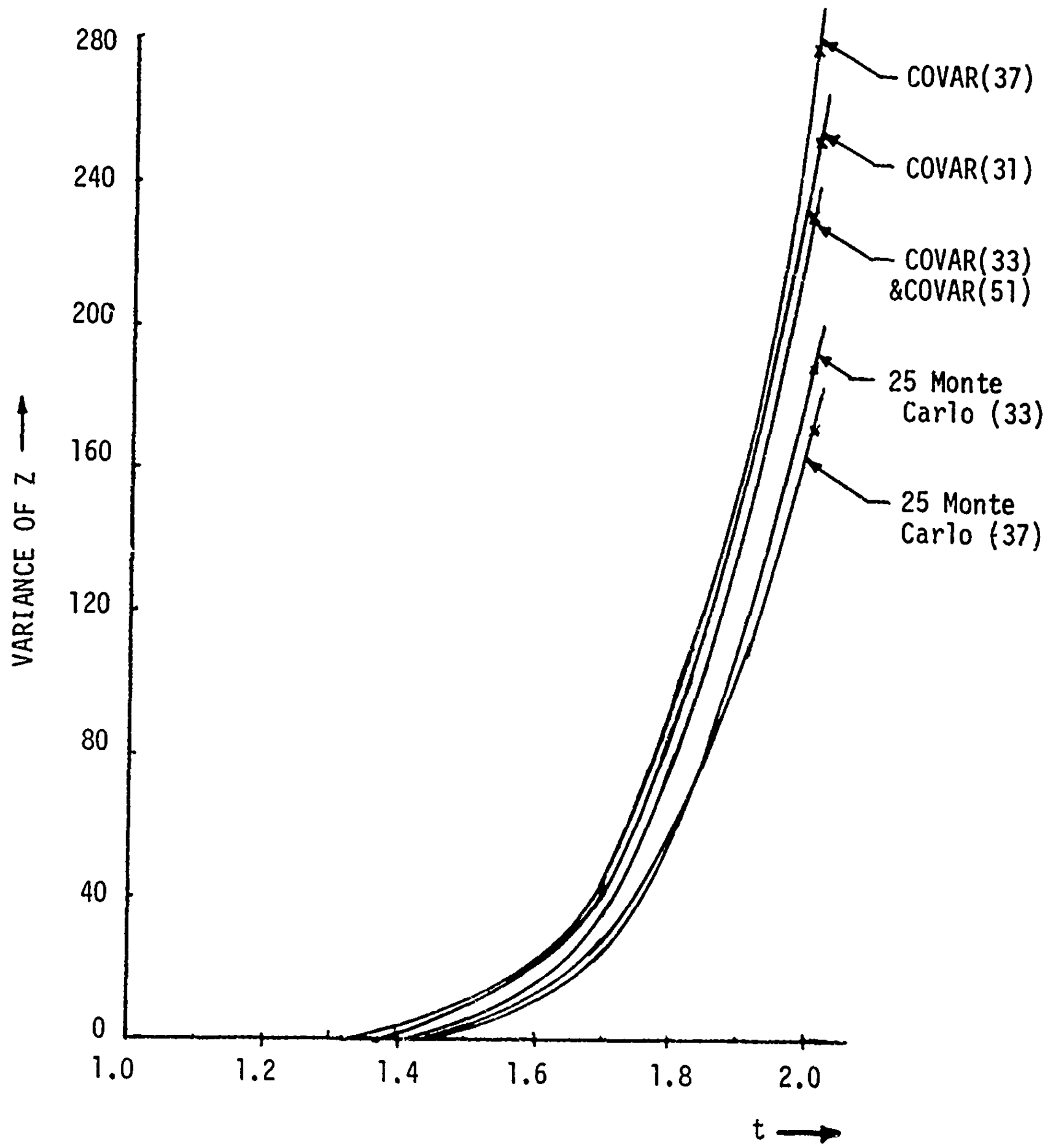


Figure 13. Numerical Comparisons Between Monte Carlo Simulations and Covariance Runs

on the range between $(0.15 \text{ degrees})^2$ and $(2.0 \text{ degrees})^2$. However noise inputs at the lower level of this range yielded poor comparisons between Monte Carlo and covariance results.

It was shown that for seeker input noise variances of $(2 \text{ degrees})^2$ the direct covariance algorithm could not be used for either that part of the flight up to one second or that part beyond twelve seconds. In those regions of operations, harsh nonlinearities prohibited the necessary linearizing assumption described in Chapter II.

Finally, the computational times and core requirements are given in Table V both for the one-to-two second interval and for the entire missile flight of approximately 12.9 seconds. These numbers are based on the assumption that the direct covariance algorithm would be used for the entire flight. Since this assumption has been shown to be invalid, these computational times will be increased for the combined computer software package described in the following section.

TABLE V

COMPUTATIONAL TIMES AND CORE REQUIREMENTS

	Computational Time (Minutes)		
System Order	Part of Flight (1.0 to 2.0 seconds)	Entire Flight (0 to 12.9 seconds)	Core Requirements (Words)
31	5.2	41	27K
33	5.6	44	28K
37	6.2	49	31K
51	9.1	72	36K

Numerical Results for the Total Flight

The combined Monte Carlo-direct covariance computer software package was run on the existing computing equipment at the U. S. Army Missile Command for the entire missile flight of 12.9 seconds. The computer run time, which included 25 Monte Carlo runs for certain portions of the flight, was approximately two and a half hours. It was shown above that both the launch segment and terminal mode of the missile flight are too nonlinear for the application of the covariance algorithm. Therefore, the sequential application of the Monte Carlo program for the first second, the covariance program for t = 1 to t = 12 seconds, and the Monte Carlo program for the final 0.9 second has been utilized to form the completed software package. The 2 1/2 hour run time for the combined program would be reduced to only approximately 45 minutes (Table V) if the missile nonlinearities had been mild enough to permit the use of the covariance algorithm on all parts of the missile flight. On the other hand, approximately 5 hours would be required for a complete Monte Carlo evaluation of 25 runs on the given system. How ·, a much larger number of runs (at least several hundred) would be needed to yield the high accuracy obtained by the covariance algorithm during the mid-portion of the flight.

Final numerical results for the total flight of approximately 12.9 seconds are given in Figures 14 and 15. Figure 14 shows the variances of X, Y, and Z as functions of time for the range on which the direct covariance algorithm is used. This curve demonstrates that the state covariance matrix elements of interest, i.e. P(19,19), P(20,20), and P(21,21), each increase monotonically on the given range. Figure

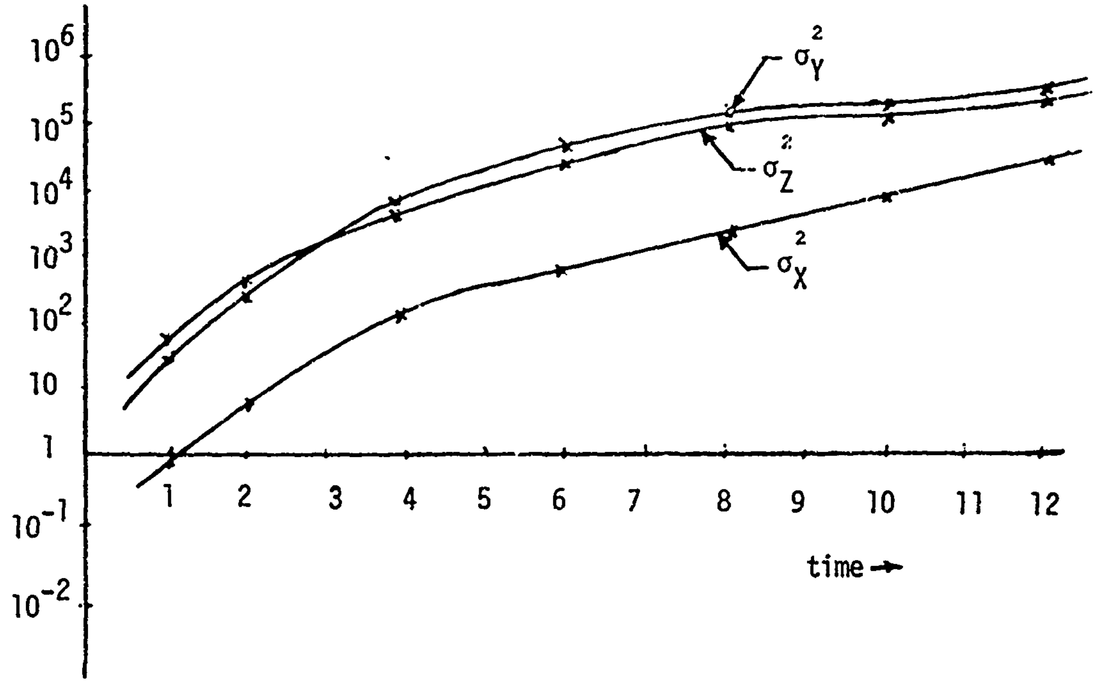


Figure 14. Position Coordinate Variance Versus Time

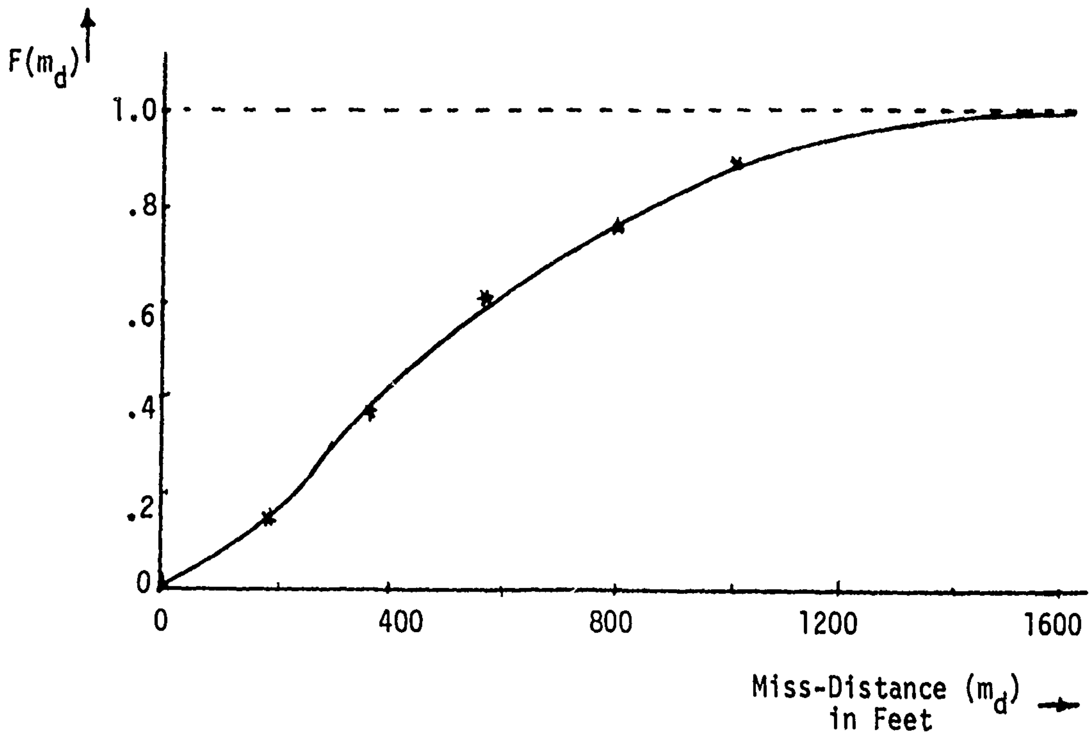


Figure 15. Probability Distribution Function of the Miss-Distance

15 shows a sketch of the probability distribution function of the miss-distance obtained from Monte Carlo runs in the terminal mode of the flight. It is apparent from Figure 15 that the seeker input noise levels of (2.0 degrees)^2 were considerably too large to yield reasonable miss-distances.

Correlated multivariate samples from a Gaussian density function equal in number to the order of the system were generated to yield the appropriate random state at t = 12 seconds for Monte Carlo simulations during the terminal mode. These samples were obtained by generating n unity variance independent Gaussian random numbers (x_i) by standard procedures. As shown by Marsaglia (30), the desired correlated random numbers (y_i) may be obtained from the triangular transformation

$$\begin{aligned}
y_1 &= g_{11}x_1 \\
y_2 &= g_{12}x_1 + g_{22}x_2 \\
y_3 &= g_{13}x_1 + g_{23}x_2 + g_{33}x_3 \\
&\vdots \\
y_k &= g_{1k}x_1 + g_{2k}x_2 + \dots \dots + g_{kk}x_k
\end{aligned} \tag{5.1}$$

where the desired covariance matrix R is used to solve for G from

$$R = GG^T \tag{5.2}$$

It can be shown (30) that the resulting elements of G satisfy

$$\begin{aligned}
g_{11} &= \sqrt{r_{11}} \\
g_{1j} &= r_{1j}/g_{11} \\
g_{ii} &= \sqrt{(r_{ii} - \sum_{m=1}^{i-1} g_{mi}^2)}, \; i > 1 \\
g_{ij} &= (r_{ij} - \sum_{m=1}^{i-1} g_{mi}g_{mj})/g_{ii}, \; j > i \\
&= 0, \; j<i
\end{aligned} \tag{5.3}$$

These calculations are included in MAIN by Cards 383-399 for the thirty-third order missile system, and the results are used in Cards 418-438.

The mildly nonlinear segment(s) of the missile flight which are amenable to solution by the direct covariance algorithm are affected by the nonlinearities themselves and the input noise levels. In particular, noise levels of (2 degrees)^2 on the seeker nonlinearities were used to obtain the results reported above. It has been shown that the region of applicability for the covariance algorithm is decreased as these noise levels are decreased. Though complete data is not available, the sketch in Figure 16 indicates typical results which one may expect. For example, levels of (0.15 degrees)^2 yielded inaccurate covariance results for the range t = 1 to t = 2 seconds. However, excellent results were obtained on this range for (2 degrees)^2, but excessive miss-distances result from such large noise levels. While the combined software package has exhibited excellent accuracy and computational speed properties for this case, its use on cases yielding acceptable miss-distances will depend on the harshness of the predominant system nonlinearities as well as the exactness of the simulation model itself.

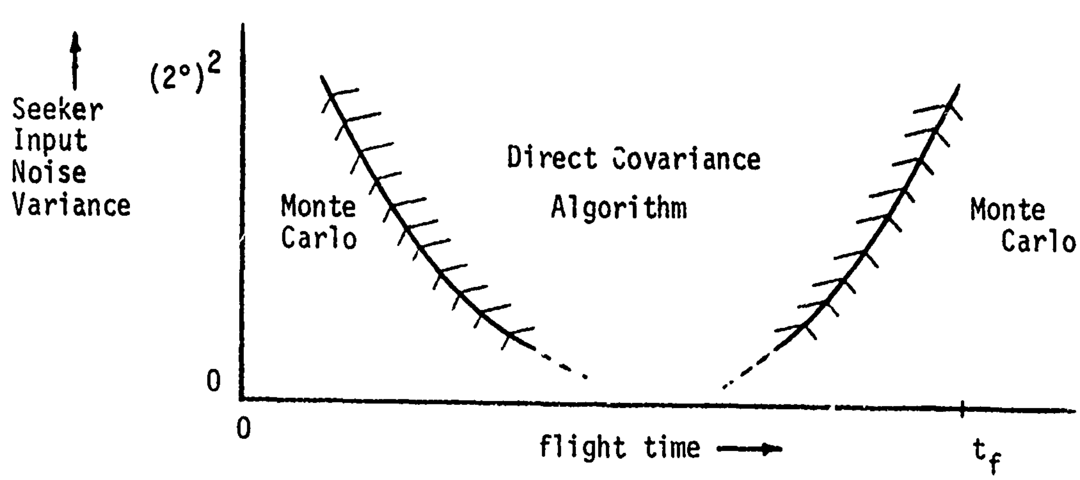


Figure 16. Sketch of a Typical Range of Applicability of the Direct Covariance Algorithm

Summary

Preliminary and final numerical results have been presented for the six degree-of-freedom air defense missile system. The direct covariance algorithm implementation was verified by comparing with 25 Monte Carlo runs on the range from $t = 1$ to $t = 2$ seconds. Thereafter, a combined computer software package was formed by using the direct covariance algorithm on the mid-portion of the flight between $t = 1$ and $t = 12$ seconds and the Monte Carlo technique on the launch and terminal parts. Finally, it was indicated that the range of applicability of the direct covariance algorithm decreased significantly for the given missile system for lower values of seeker input noise variances.

CHAPTER VI

FINAL GUIDELINES

A combined Monte Carlo-direct covariance digital computer software package for missile system analysis has been developed and tested. The completed software package is capable of handling noise propagation calculations for large scale-missile systems up to approximately 50th order. This computer program has been tailored for use on the existing Sigma 5 equipment at the U. S. Army Missile Command. In particular, the most important considerations are the resulting accuracy, computer core requirements, and program complexity. Since 48K words of core storage are presently available, the combined software package can be used without modifications for lower core requirements (Table V) on large-scale missile systems at the U. S. Army Missile Command.

Accuracy levels have been established for the six degree-of-freedom air defense system described in earlier chapters of this final report. It was shown in Chapter II that the use of only 25 Monte Carlo runs should be expected to yield errors on the order of 30% to 35%. Figure 14 in Chapter V shows that the direct covariance algorithm results differed from the results from 25 Monte Carlo simulations by approximately 30%. Therefore, the accuracy of the direct algorithm was established for the mid-portion of a typical flight. This same comparison technique indicated that Monte Carlo simulations should be used for the launch and terminal modes. Therefore, a combined Monte Carlo-direct covariance package was developed for use on a wide range of typical missile systems. Some simulation experience is needed on a given application to determine that part of the flight for which the direct covariance algorithm should be used. This experience is usually obtained during the initial simulation effort for the noise-free case.

Tradeoff possibilities with respect to accuracy, computational speed, computing equipment requirements (including storage), and program complexity were examined. It was shown that the RK2 integration formula represented an efficient tradeoff between speed and accuracy for covariance matrix calculations. The use of a general program for computing all elements of the "A" matrix was found to be inefficient. A more suitable approach involved the use of constant elements, nonlinear elements, and implicitly related elements in the proper framework. The resulting program was somewhat more complex in format, but the savings in computational time was significant.

Finally, simplifying approximations were developed to speed up the operation of the combined software package. Constant coefficients were used to replace slowly-varying elements of the "A" matrix. It was shown that during the large mid-portion of the flight, where the direct algorithm was applicable, an important approximation involved the propagation of noise through the seeker relay nonlinearities. Output variance calculations for these relays were achieved from Subroutines SNOISE and DETARA. If corresponding calculations could be performed for the large number of nonlinearities in the launch and terminal modes of flight, then the direct covariance algorithm could be utilized over a larger range of the total flight. As indicated in Figure 16 of Chapter V, the applicability of the direct covariance algorithm is also determined from the noise input levels. The proper handling of these nonlinearities will yield for given applications even greater improvements by using the combined software package.

Related Work

Comparisons between the combined software package described in this report and other approaches to noise propagation in large-scale nonlinear systems are provided in (33). Results on sensitivity analysis for noise propagation problems are included in (34). Both of these papers, as well as others, are reproduced in Appendix A of this report.

As suggested in Chapter I, an immediate extension of the noise propagation capabilities of the combined software package to filtering applications is possible. In particular, the subsequent development of an efficient software package for Kalman filtering as a practical estimation algorithm is recommended.

SELECTED BIBLIOGRAPHY

(1) RAND Corporation. A Million Random Digits with 100,000 Normal Deviates. Free Press, New York, 1955.

(2) Chambers, R. P. "Random Number Generation." IEEE Spectrum, Vol. 4, No. 2, February 1967.

(3) Hull, T. E. and A. R Dobell. "Random Number Generators." SIAM Review, Vol. 4, 1962, pp. 230-254.

(4) MacLaren, M. D. and G. Marsaglia. "Uniform Random Number Generators." Journal of the Association of Computing Machinery, Vol. 12, 1965, pp. 83-89.

(5) Gelder, A. V. "Some New Results in Pseudorandom Number Generation." Journal of the Association of Computing Machinery, Vol. 14, 1960, pp. 785-792.

(6) Box, G. E. and M. E. Muller. "A Note on the Generation of Normal Deviates." Annals of Mathematical Statistics, Vol. 28, 1958, pp. 610-611.

(7) Sage, Andrew P. Optimum Systems Control. Prentice Hall, 1968, Section 9.2, pp. 220-226.

(8) Sage, Andrew P. and James M. Melsa. Estimation Theory: With Applications to Communications and Control. New York: McGraw-Hill, 1970.

(9) Meditch, James S. Stochastic Optimal Linear Estimation and Control. New York: McGraw-Hill, 1969, Sections 4.3 and 4.4, pp. 125-152.

(10) Liebelt, Paul B. An Introduction to Optimal Estimation, Sections 4.10 and 4.16, Addison-Wesley, 1967, pp. 112-134.

(11) Bryson, Arthur E., Jr. and Yu-Chi Ho. Applied Optimal Control, Sections 11.4 and 11.5, Blaisdell Publishing Company, 1969, pp. 328-344.

(12) Jazwinski, Andrew H. Stochastic Processes and Filtering Theory, Academic Press, 1970.

(13) Kalman, R. E. and R. S. Bucy. "New Results in Linear Filtering and Prediction Theory." ASME Transactions: Journal of Basic Engineering, Vol. 83D, March 1961, pp. 95-108.

(14) Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems." ASME Transactions: Journal of Basic Engineering, Vol. 82D, March 1960, pp. 35-45.

(15) Kuhnel, Walter C. and Andrew P. Sage. "Terminal State Error Analysis Using Adjoint-Generated Sensitivities." IEEE Transactions on Aerospace and Electronic Systems, Vol. 5, No. 2, March 1969, pp. 185-194.

(16) Irwin, John D. and James C. Hung. "Methods for Injection-Error Analysis and Their Comparison." IEEE Transactions on Automatic Control, Vol. 12, No. 3, June 1967, pp. 276-281.

(17) Calfee, R. V. LTV Missiles and Space Division, "Oral Presentation to Systems Analysis Branch." U. S. Army Missile Command, Redstone Arsenal, Alabama, July, 1970.

(18) Higdon, Donald T. An Approximate Method for Determining Response of Nonlinear Dynamic Systems to Random Disturbances, Department of Aeronautical Engineering, Wichita State University, February 1967, Aeronautical Report No. 67-2, NASA Grant NGR 17-003-005.

(19) Brown, Robert J., Jr. "Trajectory Optimization for the Combined Estimation and Control of Nonlinear Stochastic Systems." School of Electrical Engg., Georgia Institute of Technology, Atlanta, Georgia, May 1971, Ph.D. Thesis.

(20) Brown, Robert J., Jr. and James R. Rowland. "Trajectory Optimization for Open-loop Nonlinear Stochastic Systems." Proceedings of the Third Annual Southeastern Symposium on System Theory, Vol. I, Georgia Institute of Technology, Atlanta, Georgia, 5-6 April 1971, Paper F4.

(21) Brown, Robert J., Jr. and James R. Rowland. "Trajectory Optimization for Closed-Loop Nonlinear Stochastic Systems." Automatic Control Conference, Washington University, St. Louis, Missouri, 11-13 August 1971, Paper No. 7-D4.

(22) Clark, George M., Jr. "Improved Estimation and Control for Nongaussian Stochastic Systems." School of Electrical Engg., Georgia Institute of Tech., Atlanta, Georgia, May 1971, Ph.D. Thesis.

(23) Clark, George M., Jr. "Specific Controller Synthesis for Linear Stochastic Systems." Proceedings of the Third Annual Symposium on System Theory, Volume II, Georgia Institute of Technology, Atlanta, Paper K6, 5-6 April 1971.

(24) Rowland, James R. and Willard M. Holmes. "Statistical Analysis Techniques for Error Propagation in Large Scale Missile Systems." Report No. RG-TR-71-19, Guidance and Control Directorate, U. S. Army Missile Command, Redstone Arsenal, Alabama, August, 1971.

(25) Rowland, James R. and Willard M. Holmes. "A Direct Covariance Algorithm for Computer-Aided Statistical Electronic Circuit Design." International Journal of Electronics, Vol. 36, No.5, May 1974.

(26) Papoulis, Athanasios. Probability, Random Variables, and Stochastic Processes, McGraw-Hill Book Company, New York, 1965.

(27) Laning, H. H. and R. H. Battin. Random Processes in Automatic Control, McGraw-Hill, 1956.

(28) Brown, Robert J., Jr., and James R. Rowland. "Autocorrelation Significance in Digital Pseudo-Random Number Generation," School of Electrical Engineering, Georgia Institute of Technology, Atlanta, Georgia, March 1970, pp. 1-20, Internal Technical Report.

(29) Rowland, James R. and Vijayendra M. Gupta. "Digital Simulations for Monte Carlo Analysis." Proceedings of the Fifteenth Midwest Symposium on Circuit Theory, University of Missouri-Rolla, Vol. I, Paper V.3, May 4-5, 1972.

(30) Marsaglia, G. "A Note on the Construction of a Multivariate Normal Sample." IRE Transactions on Information Theory, Vol. 11, No. 3, June 1957, p. 149.

(31) Rowland, James R. and Willard M. Holmes. "A Sequential Algorithm for Covariance Matrix Calculations." 1972 SWIEEECO Record of Technical Papers, Dallas, Texas, April 19-21, 1972, pp. 135-138.

(32) Rowland, James R. "Optimal Digital Simulations for Random Linear Systems with Integration Constraints." Computers and Electrical Engineering, Vol. 1, No. 1, June 1973, pp. 111-118.

(33) Gupta, Vijayendra M. and James R. Rowland. "A Survey of Direct Noise Propagation Techniques for Large-Scale Nonlinear Systems." Proceedings of the Seventeenth Midwest Symposium on Circuit Theory, Lawrence, Kansas, May 23-24, 1974.

(34) Rowland, James R. and Harold L. Pastrick. "A Stochastic Algorithm for Sensitivity Analysis." IEEE Transactions on Aerospace and Electronic Systems, In Review.

APPENDIX A

REPRINTS OF SELECTED PAPERS

This appendix contains the reprints of five selected journal and conference publications which are closely related to the work of this contract. The first of these papers, which has been listed as Reference (25), describes the application of the direct covariance algorithm to computer-aided electronic circuit analysis and design. This journal publication is based on results presented earlier in U. S. Army Technical Memorandum RG-TR-71-19 (Reference (24)). An extension of other results in Reference (24) on sequential covariance matrix calculations was presented as a conference paper at the 1972 Southwestern IEEE Conference in Dallas, Texas. This paper, listed as Reference (31), is included as the second reprint in this appendix. The third reprint, Reference (32), describes a general formulation of the optimal digital simulation problem discussed for specific cases in Chapter II of this report. A brief survey of noise propagation techniques for large-scale nonlinear systems is included as the fourth reprint (Reference (33)). Finally, the fifth paper included here describes a stochastic algorithm for sensitivity analysis. This new result (Reference (34)) provides error tolerance bounds on covariance matrix elements due to incompletely specified input noise variances.

INT. J. ELECTRONICS, Vol. 36, No. 5, May 1974.

A direct covariance algorithm for computer-aided statistical electronic circuit design

JAMES R. ROWLAND

School of Electrical Engineering and Center for Systems Science, Oklahoma State University, Stillwater, Oklahoma 74074

and WILLARD M. HOLMES

Guidance and Control Directorate (AMSMI-RGN), Research, Development, Engineering and Missile Systems Laboratory, U.S. Army Missile Command, Redstone Arsenal, Alabama 35809

[Received 21 May 1973]

A direct covariance algorithm is presented for handling problems of component tolerance analysis and random input variations with a particular emphasis for utilization in computer-aided statistical electronic circuit design. It is shown that this result is applicable to a wide range of electronic circuit arrays having non-linear components. Moreover, a systematic procedure is developed for predicting in advance the expected accuracy. Numerical results comparing the direct covariance algorithm with up to 1000 Monte Carlo ensemble-averaged computer runs are provided. Contrary to popular belief, errors of 10 to 25% are obtained by using 25 to 100 Monte Carlo runs. Improvements in both accuracy and computational speed clearly demonstrate that the direct covariance algorithm is a versatile and effective computer-aided design tool.

1. Introduction

Noise problems inherent in practical circuit designs are frequently identified only after the basic design has been completed and production testing has begun. Rarely do statistical performance design requirements proceed parallel with other design requirements. A first step in establishing these statistical design requirements is the development of a fast, effective statistical analysis tool for use during the preliminary design. While the traditional Monte Carlo method provides acceptable statistical results by using a sufficiently large number of digital simulation runs, its frequent use during the design stage can become prohibitively expensive. As a circuit array increases in size and complexity, digital computer time for a single simulation run goes up very rapidly. Repeated runs further increase the computational time and associated computer costs. An efficient, easily applied, statistical analysis technique having a reliable accuracy is needed to pinpoint potential noise problems during the developmental stages of electronic circuit design.

The increasing emphasis on statistical analysis techniques in computer-aided circuit design has resulted in expanded programmes for handling problems in component tolerance analysis, modelling, and simulation. For example, an extensive continuing programme in computer-aided statistical circuit design has been described by Dickieson and Chernak (1971). Semmelman *et al.* (1971) and Cermak and Kirby (1971) have discussed present state-of-the-art capabilities for linear and non-linear computer-aided statistical circuit design.

Furthermore, Logan (1971) described the characterization and modelling of components for tolerance analysis, and Karafin (1971) used tolerance analysis for optimum design. More recently, Pinel and Roberts (1972) treated the tolerance assignment problem for linear networks on a worst-case basis by non-linear programming.

This paper uses the state-space approach, described for circuit analysis and design by Pottle (1966) and Yarlagadda (1972), to develop a direct covariance algorithm for determining the effects resulting from random input and/or component variations. Related results by Irwin and Hung (1967), Kuhnel and Sage (1969), and Rowland and Holmes (1971) have been used for large-scale, non-linear systems in aerospace applications. These results were based on earlier work in linear filtering theory by Kalman (1960). The contributions of this paper are (1) the development and application of the direct covariance algorithm for linear and non-linear circuit analysis problems, (2) the development of an accuracy prediction scheme for estimating in advance the range of applicability in non-linear cases, and (3) numerical comparisons showing the need for a very large number of Monte Carlo runs for comparable accuracy.

2. The direct covariance algorithm

Consider a non-linear circuit whose dynamical response may be expressed in state variable form as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{r}(t), \mathbf{w}(t), \boldsymbol{\alpha}, t) \tag{1}$$

where $\mathbf{x}$ is an n-dimensional vector representing the circuit state, $\mathbf{r}(t)$ is a k vector of non-random inputs, $\mathbf{w}(t)$ is an m vector of random process circuit inputs and/or parameters, and $\boldsymbol{\alpha}$ is a j vector of random bias (i.e. random variable) circuit inputs and/or parameters. As indicated, the n vector $\mathbf{f}$ is a non-linear functional of those vector arguments shown in eqn. (1).

Let the mean values of $\mathbf{w}(t)$ and $\boldsymbol{\alpha}$ be represented by η_w and η_α, respectively. Observe that the distinction between the random vectors $\mathbf{w}(t)$ and $\boldsymbol{\alpha}$ is that $\mathbf{w}(t)$ is a white noise random process while $\boldsymbol{\alpha}$ is a random variable that is constant in time. Let the covariance matrices of $\mathbf{w}(t)$ and $\boldsymbol{\alpha}$ be defined by

$$\left.\begin{aligned} E\{(\mathbf{w}(t)-\eta_w(t))(w(\tau)-\eta_w(\tau))^T\} &\triangleq Q_w(t)\delta(t-\tau) \\ E\{(\boldsymbol{\alpha}-\eta_\alpha)(\boldsymbol{\alpha}-\eta_\alpha)^T\} &\triangleq Q_\alpha \end{aligned}\right\} \tag{2}$$

where $\delta(\cdot)$ represents the delta function.

It is assumed that $\mathbf{f}$ in eqn. (1) is a sufficiently smooth functional of its arguments such that its first partial derivatives with respect to $\mathbf{x}$, $\mathbf{w}(t)$ and $\boldsymbol{\alpha}$ exist. Let $\mathbf{f}$ be expanded in a Taylor series about the noise-free solution $\mathbf{x}_N(t)$ to yield from eqn. (1) the linearized incremental equation given by

$$\delta\dot{\mathbf{x}} = A(t)\delta\mathbf{x} + B(t)\delta\mathbf{w}(t) + C(t)\delta\boldsymbol{\alpha} \tag{3}$$

where the noise-free solution is the solution of eqn. (1) obtained by replacing the noise vectors $\mathbf{w}(t)$ and $\boldsymbol{\alpha}$ by their mean values, i.e.

$$\dot{\mathbf{x}}_N(t) = \mathbf{f}(\mathbf{x}_N, \mathbf{r}(t), \eta_w, \eta_\alpha, t) \tag{4}$$

Moreover, the matrices $A(t)$, $B(t)$ and $C(t)$ in eqn. (3) are used to represent first partial derivatives defined by

$$\left.\begin{aligned} A(t) &\triangleq \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_N \\ B(t) &\triangleq \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}}\right|_N \\ C(t) &\triangleq \left.\frac{\partial \mathbf{f}}{\partial \boldsymbol{\alpha}}\right|_N \end{aligned}\right\} \tag{5}$$

where the subscript N is used to denote that the partial derivatives are evaluated at the nominal, or noise-free, condition. Finally, the incremental variations in $\mathbf{x}$, $\mathbf{w}(t)$ and $\boldsymbol{\alpha}$ about their nominal values are given by

$$\left.\begin{aligned} \delta\mathbf{x} &\triangleq \mathbf{x}(t) - \mathbf{x}_N(t) \\ \delta\mathbf{w}(t) &\triangleq \mathbf{w}(t) - \eta_w(t) \\ \delta\boldsymbol{\alpha} &\triangleq \boldsymbol{\alpha} - \eta_\alpha \end{aligned}\right\} \tag{6}$$

It is assumed that these incremental variations are sufficiently small such that second and higher-order Taylor series terms in eqn. (3) may be neglected.

The statistical analysis problem under consideration is to determine the state covariance matrix $P(t)$ which results from the presence of random vectors $\mathbf{w}(t)$ and $\boldsymbol{\alpha}$ in the dynamical eqn. (1) of the particular circuit. It is shown in the Appendix that $P(t)$ satisfies the matrix differential equation given by

$$\begin{aligned} (\dot{P}t) = A(t)P(t) + P(t)A^T(t) + B(t)Q_w(t)B^T(t) \\ + C(t)Q_\alpha H^T(t) + H(t)Q_\alpha C^T(t) \end{aligned} \tag{7}$$

where

$$\left.\begin{aligned} P(t) &\triangleq E\{\delta\mathbf{x}\delta\mathbf{x}^T\} \\ H(t) &\triangleq \int_0^t \Phi(t, \tau)C(\tau)\,d\tau \end{aligned}\right\} \tag{8}$$

and $\Phi(t, \tau)$ is the state transition matrix associated with $\delta\mathbf{x}$ in eqn. (3).

The matrix equation in eqn. (7) is exact for the linear, time-varying incremental equation for $\delta\mathbf{x}$ in eqn. (3). However, since second and higher-order terms in the Taylor series expansion of $\mathbf{f}$ have been neglected in arriving at eqn. (3), the application of the direct covariance result in eqn. (7) must be recognized as providing only an approximate analysis for the non-linear dynamical circuit in eqn. (1). Particular examples described in the following section demonstrate that the linearization assumption is justified for low-noise, mildly non-linear circuits.

3. Numerical results

Two examples are presented here to illustrate the usefulness of the direct covariance algorithm for circuit analysis as well as to indicate its limitations in certain highly non-linear cases. Following a brief first example involving a simple RL series circuit with R being treated as a random variable, comparisons with the Monte Carlo approach are made for a non-linear, second-order, cascaded RC ladder circuit. The need for ensemble-averaging a very large number of Monte Carlo simulation runs for comparable accuracy is demonstrated, and the resulting advantages of the direct covariance approach are clearly identified.

Example 1

Let the resistance R in a simple RL series circuit be represented as a random variable that is uniformly distributed on the range between $\eta_R - R_0$ and $\eta_R + R_0$, where η_R is 10 ohms and R_0 is allowed to assume several constant values for purposes of comparison. Elementary considerations may be used to show that the variance of R is related to the bounds on the probability density function by $Q_R = R_0^2/3$. Moreover, let the source be a d.c. voltage of magnitude $V_s = 100$ volts applied for all $t \geq 0$, and let L be 100 millihenrys.

The voltage v_R across the resistor, initially zero, obeys the scalar dynamical circuit equation given by

$$\dot{v}_R = -\frac{R}{L} v_R + \frac{R}{L} V_s \tag{9}$$

with a noise-free solution defined by

$$V_{RN}(t) = V_s[1 - \exp(-\eta_R t/L)] \tag{10}$$

The linearized incremental equation corresponding to eqn. (3) is

$$\delta\dot{v}_R = \left(-\frac{\eta_R}{L}\right)\delta v_c - \frac{1}{L}(v_{RN}(t) - V_s)\delta R \tag{11}$$

Even though the series RL circuit itself is linear, the appearance of the term with R in eqn. (9) as a product with v_R forces the problem into a general non-linear framework and requires the usual linearization assumption of sufficiently small variations.

Inserting eqn. (10) into eqn. (11) and identifying the system coefficient matrices in eqn. (3) yields the covariance matrix differential equation from eqns. (7) and (8) as

$$\dot{P}(t) = -\frac{2\eta_R}{L} P(t) + \frac{2V_s^2 R_0^2 t}{3L^2} \exp(-2\eta_R t/L) \tag{12}$$

which has the closed-form solution (for $P(0) = 0$) given by

$$P(t) = \frac{V_s^2 R_0^2}{3L^2} t^2 \exp(-2\eta_R t/L) \tag{13}$$

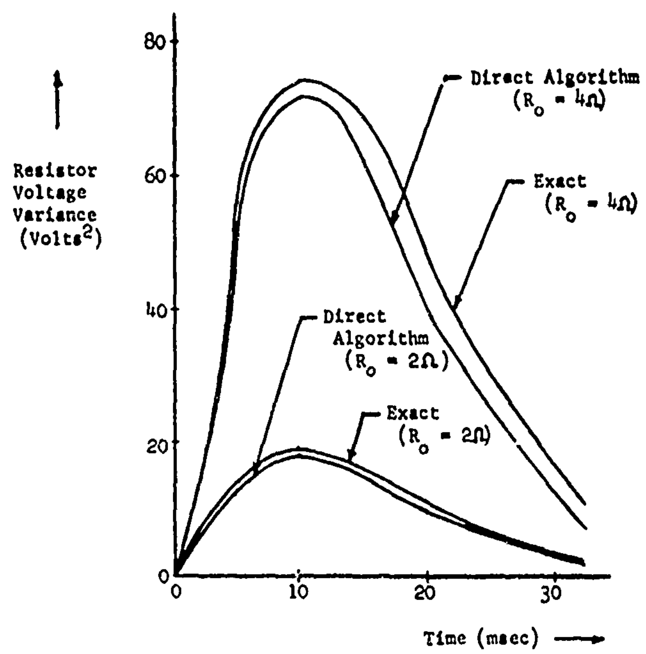
Using eqns. (9) and (10) with basic definitions from probability theory provides the exact solution $P_{ex}(t)$ for the variance of the voltage across the resistor as

$$P_{ex}(t) = \int_{\eta_R - R_0}^{\eta_R + R_0} \left[V_s[1 - \exp(-Rt/L)] - \int_{\eta_R - R_0}^{\eta_R + R_0} V_s[1 - \exp(-\rho t/L)] \times \left(\frac{1}{2R_0}\right) d\rho \right]^2 \left(\frac{1}{2R_0}\right) dR$$

$$= V_s^2 \exp(-2\eta_R t/L) \left[\frac{\exp(2R_0t/L) - \exp(-2R_0t/L}{4R_0t/L} - \left(\frac{\exp(R_0t/L) - \exp(-R_0t/L)}{2R_0t/L} \right)^2 \right] \quad (14)$$

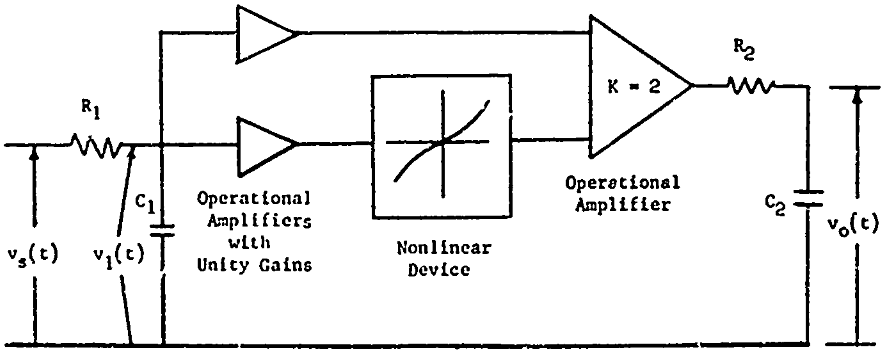
Comparisons between this exact solution and the approximate result in eqn. (13) from the direct covariance approach are presented in fig. 1 for the given conditions. These two solutions differ only slightly for rather wide ranges of R_0 for this mildly non-linear application of the direct covariance algorithm. Furthermore, the large magnitudes obtained in fig. 1 for the resistor voltage variances indicate that close parameter tolerances can be quite important in circuit design considerations.

Fig. 1



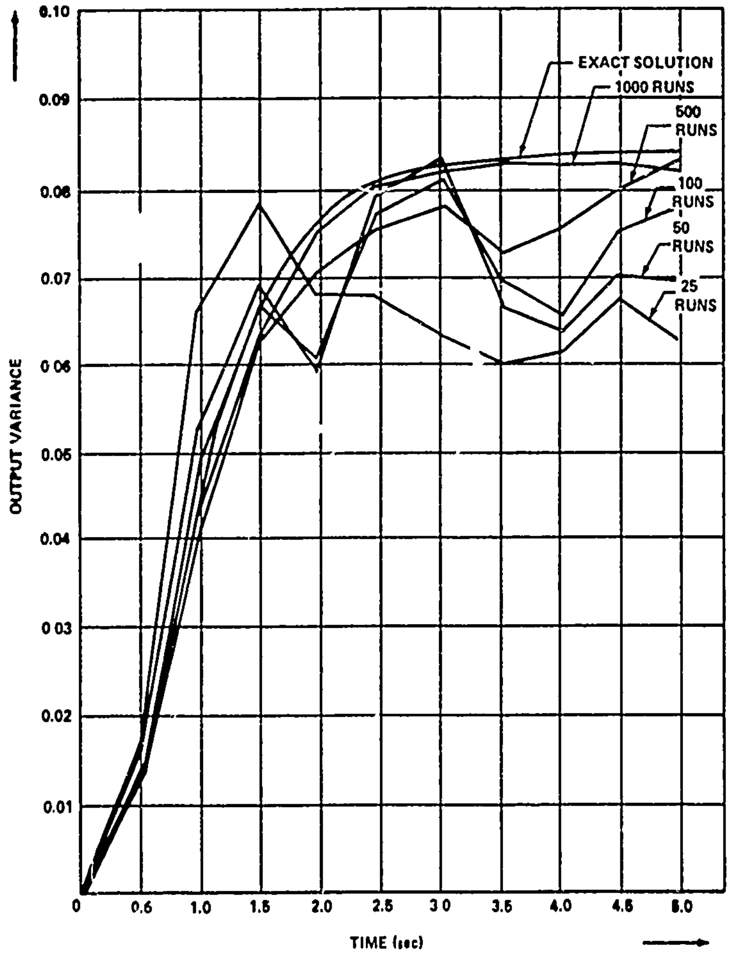
Comparisons between the direct covariance algorithm and the exact solution showing the variance of the resistor voltage for Example 1.

Fig. 2



The second-order non-linear electronic circuit considered in Example 2.

Fig. 3



Monte Carlo results for the linear case ($\gamma = 0$) of the circuit in fig. 2.

Example 2

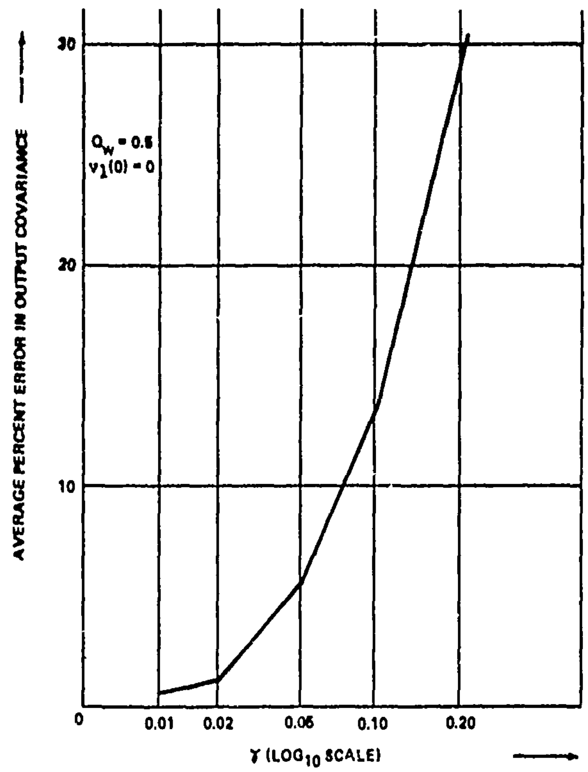
Consider the second-order non-linear circuit shown in fig. 2 and represented dynamically by

$$\left.\begin{aligned} \dot{v}_1 &= -\frac{1}{R_1C_1}v_1+\frac{1}{R_1C_1}v_s(t) \\ \dot{v}_0 &= -\frac{1}{R_2C_2}v_0+\frac{K}{R_2C_2}v_1+\frac{K\gamma}{R_2C_2}v_1|v_1| \end{aligned}\right\} \tag{15}$$

where $R_1C_1=1$, $R_2C_2=\frac{1}{2}$, and the source $v_s(t)$, applied for all $t \geqslant 0$, is a zero-mean Gaussian white noise input with variance Q_w. The operational amplifiers are included for amplification, isolation, and summing. The initial voltage on C_2 is zero, but $v_1(0)$, Q_w and the constant scalar parameter γ are allowed to assume different values as indicated below.

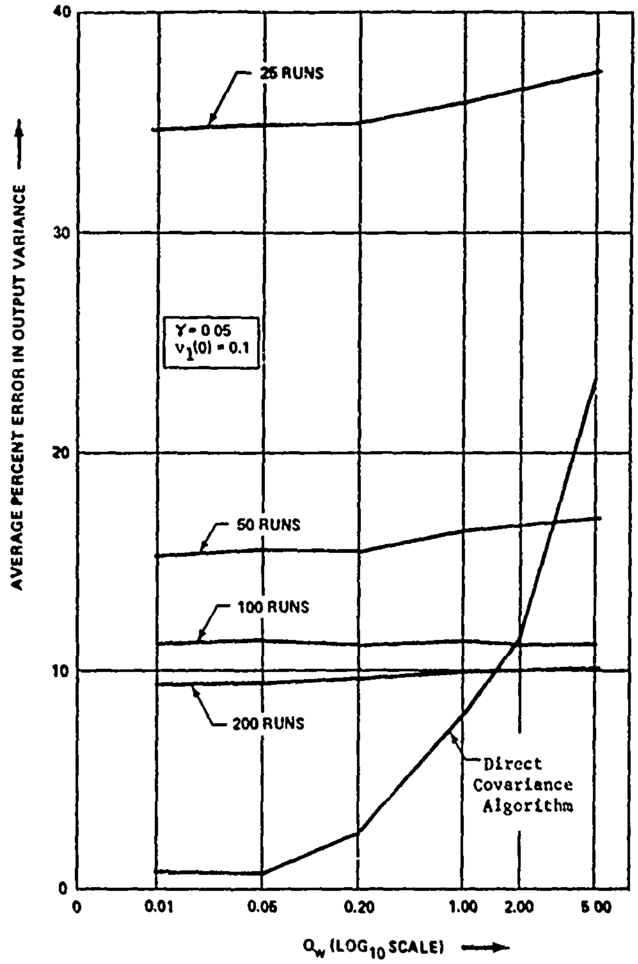
The purpose of this example is to present comparisons with Monte Carlo simulation runs and to demonstrate the range of applicability of the direct covariance algorithm for non-linear electronic circuit analysis. Figure 3 shows curves of ensemble-averaged Monte Carlo runs performed on the digital computer for the linear case ($\gamma=0$) with $Q_w=1$ and $v_1(0)=0$. The variance

Fig. 4



Variations in average per cent error in the output voltage variance versus γ for the direct covariance algorithm applied to Example 2.

Fig. 5



Digital simulation results for the direct covariance algorithm and the Monte Carlo technique as Q_w varies in Example 2.

of the output voltage $v_0(t)$, which is plotted as a function of time during the transient region of operation, exhibits errors of from 10 to 25% for 25 to 100 Monte Carlo runs. Comparisons with the exact solution obtained by using the direct covariance algorithm reveals that up to 1000 Monte Carlo runs are needed for approximately 2% accuracy.

Variations in direct covariance results as a function of the amount (γ) of circuit non-linearity are illustrated in fig. 4. For the same time period as in fig. 3, but with $Q_w = 0{\cdot}5$ and $v_1(0) = 0$, the average per cent error in the output voltage variance is plotted versus γ. A similar result is shown in fig. 5 for variations in Q_w with $\gamma = 0{\cdot}05$ and $v_1(0) = 0{\cdot}10$. These computer simulation runs demonstrate that the error in the direct covariance solution, when compared with 1000 Monte Carlo runs, increased as γ and Q_w increased and, consequently, as the given electronic circuit became more non-linear. Both curves are used in the following section to estimate the accuracy expected from the direct covariance algorithm by examining the non-linear circuital equations directly. Moreover, fig. 5 indicates not only that this approximate algorithm might be unacceptable for highly non-linear circuits but also re-emphasizes the earlier result that a very large number of Monte Carlo runs are required to obtain accurate results.

4. Accuracy prediction

It would be desirable to be able to predict in advance the accuracy of the direct covariance algorithm for non-linear circuits. An exact prediction of the expected accuracy is not possible because exact analytical solutions cannot be found in general, for the output variance of non-linear circuits. However, the result from a large number of Monte Carlo runs may be regarded as a reference solution for the purpose of accuracy prediction, but even then (as shown in fig. 3) some inaccuracy is present. The reason for using the direct covariance technique is to avoid the time-consuming Monte Carlo approach.

Suppose the Monte Carlo runs had been made for one particular design condition (parameter setting) of a given electronic circuit. Using this information, the following procedure could be used to estimate the accuracy of the direct covariance algorithm for sufficiently small changes in the parameter settings. As a particular example to illustrate the procedure, consider the exact incremental equation associated with eqn. (15), i.e.

$$\left.\begin{aligned} \delta\dot{v}_1 &= -\frac{1}{R_1C_1}\,\delta v_1 = \frac{1}{R_1C_1}\,v_s(t) \\ \delta\dot{v}_0 &= -\frac{1}{R_2C_2}\,\delta v_0 + \frac{K}{R_2C_2}\,[1+2\gamma|v_1|]_N\delta v_1 + \frac{K\gamma}{R_2C_2}\,\delta v_1^2 \end{aligned}\right\} \tag{16}$$

Suppose that the non-linear term in eqn. (16) is required to be not greater than $k\%$ of the corresponding linear terms, i.e.

$$|\gamma\delta v_1^2| \leqslant \frac{k}{100}\,|-2\delta v_0 + [1+2\gamma|v_1|]_N\delta v_1| \tag{17}$$

where $K=2$ and $R_2C_2=\frac{1}{2}$ have been substituted into eqn. (17). Squaring and taking expected values yields

$$\gamma^2(3\sigma_{\delta v_1}^4) \leqslant \left(\frac{k}{100}\right)^2 [4\sigma_{\delta v_0}^2 + [1+2\gamma|v_1|]_N^2\,\sigma_{\delta v_1}^2 + 4[1+2\gamma|v_1|]_N\,|E\{\delta v_0\delta v_1\}|] \tag{18}$$

Note that $E\{\delta v_1^4\}$ has been approximated by $3\sigma_{\delta v_1}^4$, which is exact in this case because δv_1 is Gaussian. Using the steady-state values of the variance terms obtained from the linear case ($\gamma=0$) yields

$$\sigma_{\delta v_0}^2 = \frac{Q_w}{12}\,; \quad \sigma_{\delta v_1}^2 = \frac{Q_w}{2}\,; \quad \Gamma\{\delta v_0\delta v_1\} = \frac{Q_w}{6} \tag{19}$$

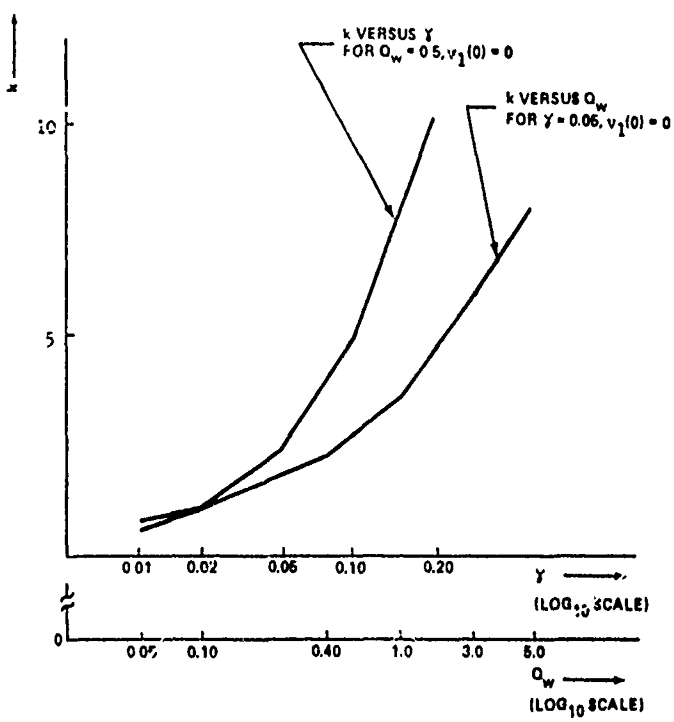
Substituting eqn. (19) into eqn. (18) gives, after simplifications,

$$3\gamma^2 Q_w \leqslant \left(\frac{k}{100}\right)^2 \left[\tfrac{1}{3} + \tfrac{2}{3}\{1 + 2\gamma v_1(t)\}^4 - 2\{1 + 2\gamma v_1(t)\}^2\right] \tag{20}$$

The equality in eqn. (20) is plotted in fig. 6, which shows that as either γ or Q_w increases, the per cent k of the second incremental equation in eqn. (16) caused by the non-linear term increases rapidly. Using the information in fig. 6 together with figs. 4 and 5, the per cent error in the output variance as a function of the parameter k may be plotted. The sketch for varying γ and Q_w is shown in fig. 7. If k is less than 3%, then the error in the output variance is less than 5%. However, for $k = 10\%$, the error in the output variance is approximately 30%. If γ and Q_w are such that k is approximately 10%, then the direct covariance algorithm compares in accuracy to approximately 25 Monte Carlo runs (30% error). However, if $k = 3\%$, then the accuracy of the direct covariance algorithm is better than 200 Monte Carlo runs. Therefore, k may be computed in advance from the incremental equations to determine the expected accuracy and the number of Monte Carlo runs which would yield approximately the same accuracy as the direct covariance algorithm for the given non-linear circuit.

These observations on the accuracy of the direct covariance algorithm as a function of the quantity k are precisely correct only for the single example

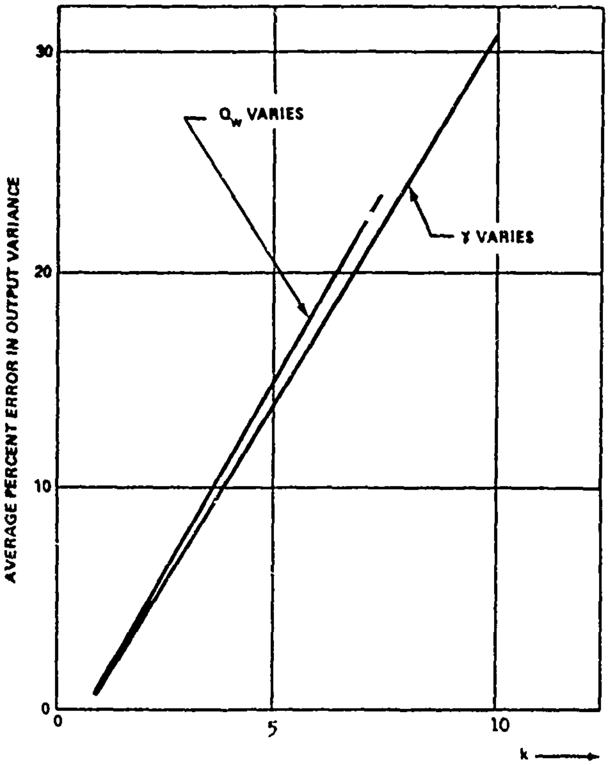
Fig 6



Plots of k versus γ and Q_w for the non-linear circuit in eqn. (16).

Covariance algorithm for computer aided electronic circuit

Fig. 7



Plots of per cent error for the direct covariance algorithm versus k for the circuit described by eqn. (15).

considered. However, it can be expected that other similar second-order circuits with parameters sufficiently near those of the previous example would yield results with corresponding accuracy. In particular, it should be expected, as shown in fig. 7, that the average per cent error in output variance would be on the order of threes time the value of k. Moreover, some useful information would be obtained even if this error varied by as much as two to four times k. However, variations of from 50 to 100 times k would be unexpected.

Monte Carlo simulation experience is usually available on those electronic circuits where noise disturbances have been a problem. Curves similar to those in fig. 7 can be plotted for the particular non-linear circuit being considered. As stated previously, these curves can be used to yield approximate estimates of the accuracy of the direct covariance algorithm in given situations

5. Software package development

A digital computer software package for implementing the direct covariance algorithm for large-scale circuits and systems has been developed.

The completed package provides the capability for statistical analyses with realistic engineering trade-offs between accuracy, computational speed, equipment requirements, and programme complexity for the user. The importance of such a computer software package for electronic circuit analysis is evident from its potential usage for parallel statistical analysis during preliminary design. The evolutionary nature of this statistical information tends to minimize the need for redesign during terminal stages of circuit development, which was discussed by Dawson *et al.* (1966).

A major consideration for the use of the direct covariance package in circuit design is its inherent computational efficiency. While comparable accuracy from the Monte Carlo approach requires up to $1000n$ system integrations, where n is the order of the system or circuit being designed, the direct covariance algorithm requires $n(n+1)/2$ such system integrations. If n is 50, for example, the direct covariance algorithm operates approximately 40 times fast than the Monte Carlo approach. On the other hand, for very large circuit arrays of extremely high order, the relative economy between the two statistical analysis tools diminishes. However, as shown in a previous section, extreme cases of circuits involving high noise sources and/or very harsh non-linearities should be handled by the traditional Monte Carlo method.

6. Conclusions

A direct covariance algorithm has been developed and applied to the circuit analysis problem for utilization as part of a general computer-aided statistical analysis and design capability. The advantages in both computational speed and accuracy over the traditional Monte Carlo technique have been demonstrated for low-noise, mildly non-linear circuits. In addition, a procedure for accuracy prediction has been developed and applied to a typical example. The incorporation of this direct covariance algorithm into a digital computer software package has been described with particular emphasis on its importance to the user as a circuit analysis tool for preliminary statistical design.

Appendix

This Appendix presents the derivation of the covariance matrix differential equation in eqn. (7) for the linear incremental equation in eqn. (3). The exact solution for $\delta\mathbf{x}(t)$ may be expressed in terms of its state transition matrix $\Phi(t, t_0)$ as

$$\delta\mathbf{x}(t)=\Phi(t, t_0)\delta\mathbf{x}(t_0)+\int_{t_0}^{t}\Phi(t, \tau)B(\tau)\delta\mathbf{w}(\tau)\,d\tau+\int_{t_0}^{t}\Phi(t, \tau)C(\tau)\delta\boldsymbol{\alpha}\,d\tau \tag{21}$$

Recognizing that $\delta\boldsymbol{\alpha}$ through random, is constant in time and using the definition of $H(t)$ from eqn. (8), one has

$$\delta\mathbf{x}(t)=\Phi(t, t_0)\delta\mathbf{x}(t_0)+\int_{t_0}^{t}\Phi(t, \tau)B(\tau)\delta\mathbf{w}(\tau)\,dt+H(t)\delta\boldsymbol{\alpha}$$

Therefore,

$$P(t) = E\{\delta\mathbf{x}(t)\delta\mathbf{x}^{\mathrm{T}}(t)\}$$

$$= E\left[\Phi(t, t_0)\delta\mathbf{x}(t_0) + \int_{t_0}^{t} \Phi(t, \tau)B(\tau)\delta\mathbf{w}(\tau)\, d\tau + H(t)\delta\boldsymbol{\alpha}\right]$$

$$\times\left[\Phi(t, t_0)\delta\mathbf{x}(t_0) + \int_{t_0}^{t} \Phi(t, \tau)B(\tau)\delta\mathbf{w}(\tau)\, d\tau + H(t)\delta\boldsymbol{\alpha}\right]^{\mathrm{T}} \tag{23}$$

Performing the indicated multiplications in eqn. (23) and noting that $\delta\mathbf{x}(t_0)$, $\delta\mathbf{w}(t)$ and $\delta\boldsymbol{\alpha}$ are uncorrelated yields the result

$$P(t) = \Phi(t, t_0)E[\delta\mathbf{x}(t_0)\delta\mathbf{x}^{\mathrm{T}}(t_0)]\Phi^{\mathrm{T}}(t, t_0)$$

$$+ \int_{t_0}^{t}\int_{t_0}^{t} \Phi(t, \tau)B(\tau)E[\delta\mathbf{w}(\tau)\delta\mathbf{w}^{\mathrm{T}}(\rho)]B^{\mathrm{T}}(\rho)\Phi^{\mathrm{T}}(t, \rho)\, d\tau\, d\rho$$

$$+ H(t)E\{\delta\boldsymbol{\alpha}\delta\boldsymbol{\alpha}^{\mathrm{T}}\}H^{\mathrm{T}}(t) \tag{24}$$

Using eqn. (2) and the sifting property of the delta function, one obtains

$$P(t) = \Phi(t, t_0)P(t_0)\Phi^{\mathrm{T}}(t, t_0)$$

$$+ \int_{t_0}^{t} \Phi(t, \tau)B(\tau)Q_{\mathrm{w}}(\tau)B^{\mathrm{T}}(\tau)\Phi^{\mathrm{T}}(t, \tau)\, d\tau$$

$$+ H(t)Q_{\alpha}H^{\mathrm{T}}(t) \tag{25}$$

Equation (25) provides the integral solution for $P(t)$. However, by forming $\dot{P}(t)$ from eqn. (25) and using the relationship

$$\frac{\partial\Phi(t, \tau)}{\partial t} = A(t)\Phi(t, \tau) \tag{26}$$

the direct covariance algorithm may be expressed in the more convenient form of the matrix differential equation in eqn. (7).

References

CERMAK, I. A., and KIRBY, D. B., 1971, *Bell Syst. tech. J.*, **50,** 1173.
DAWSON, D. F., KUO, F. F., and MAGNUSON, W. G., 1966 *Proc. I.E.E.E.*, **55,** 1946.
DIKIESON, A. C., and CHERNAK, J., 1971, *Bell Syst. tech. J.*, **50,** 1099.
IRWIN, J. D., and HUNG, J. C., 1967, *I.E.E.E. Trans. autom. Control*, **12,** 276.
KALMAN, R. E., 1960, *J. bas. Engng*, D, **82,** 35.
KARAFIN, B. J., 1971, *Bell Syst. tech. J.*, **50,** 1225.
KUHNEL, W. C., and SAGE, A. P., 1969, *I.E.E.E. Trans. Aerospace Electron. Syst.*, **5,** 185.
LOGAN, J., 1971, *Bell Syst. tech. J.*, **50,** 1105.
PINEL, J. F., and ROBERTS, K. A., 1972, *I.E.E.E. Trans. Circuit Theory*, **19,** 475.
POTTLE, C., 1966, *System Analysis by Digital Computer* (New York : Wiley).
ROWLAND, J. R., and HOLMES, W. M., 1971, Tech. Rep. No. RG–TR–71–19, U.S. Army Missile Command, Redstone Arsenal, Ala.
SEMMELMAN, C. L., WALSH, E. D., and DARYANANI, G. T., 1971, *Bell Syst. tech. J.*, **50,** 1149.
YARLAGADDA, R., 1972, *I.E.E.E. Trans. Circuit Theory*, **19,** 227.

A SEQUENTIAL ALGORITHM FOR COVARIANCE MATRIX CALCULATIONS

James R. Rowland
School of Electrical Engineering
and Center for Systems Science
Oklahoma State University
Stillwater, Oklahoma 74074

Willard M. Holmes
Guidance and Control Directorate (AMSMI-RGN)
Research, Development, Engineering and
Missile Systems Laboratory
U. S. Army Missile Command
Redstone Arsenal, Alabama 35809

ABSTRACT

A useful sequential algorithm is developed for handling state covariance matrix calculations in large-scale stochastic filtering and analysis problems. A significant reduction in computer storage is obtained by segmenting large-scale systems and operating sequentially on the various subsystems. This saving in computer storage is due to a procedure of dimensioning intermediate integration variables on a lower-order subsystem basis and re-using them from subsystem to subsystem. Error considerations and the amount of core reduction achieved are discussed, and an example is presented to illustrate the sequential ordering of the covariance matrix calculations.

1. INTRODUCTION

It is important to be able to perform computations sequentially for large-scale systems to avoid excessive digital computer storage requirements. For example, such considerations are especially critical in large-scale air defense missile system simulations where a variety of operations must be handled simultaneously [1,2]. Some of these systems are so large that is is simply not possible to implement the desired stochastic filtering or analysis algorithm directly on a given computing facility. In such cases, an approximate method must be used.

The sequential algorithm developed in this paper is based on the multilevel systems concept proposed by Mesarovic [3], who partitioned complex systems into simpler subsystems to form a hierarchy of system models for analysis and design purposes. In [4] Lefkowitz described how the multilevel hierarchy approach had been used to solve particular industrial problems. Moreover, Noton [5] applied multilevel systems theory to derive a coordination algorithm for a number of subsystem Kalman estimators.

In the present work, the overall system is segmented into several subsystems interconnected by feedforward and feedback paths. The analysis problem considered is the evaluation of the state covariance matrix at discrete points in time from its matrix differential equation. Using the subsystem concept, one may partition the coefficient matrices, the input covariance matrix, and the state covariance matrix to permit simplified sequential calculations. Differential equations for these partitioned segments are written to reflect self-interacting, feedforward, feedback, and input terms. The numerical integration of these subsystem covariance matrix differential equations is performed sequentially on the digital computer with a worthwhile savings in computer storage. Results from a given subsystem calculation become a part of the forcing functions for connected subsystems considered subsequently. The reason for the lower required storage is that intermediate integration variables are dimensioned on a lower-order subsystem basis and re-used from subsystem to subsystem. Moreover, some subsystems have inputs from only a few other subsystems, which further simplifies the sequential computations. On the other hand, integrating the system covariance matrix differential equation in its original form requires much larger dimension statements for the intermediate variables. The reduction in core storage is a function of the number, order, and arrangement of subsystems, including the various interconnections of feedforward and feedback loops.

The cost of obtaining the reduction in core storage requirements is reflected in the increased complexity involved in the ordering of calculations for the sequential algorithm. For those applications where less computational accuracy is acceptable, additional savings in computer storage and/or computational speed can be realized. An example is presented to illustrate the sequential algorithm itself as well as the interesting tradeoffs possible in its implementation for large-scale systems.

2. THE SEQUENTIAL ALGORITHM

Consider a linear, time-varying system described by the vector differential equation

$$\dot{\underline{x}} = A(t)\underline{x} + B(t)\underline{w} \qquad (1)$$

where $\underline{x}$ is the n-vector of system states, $\underline{w}$ is an ℓ-vector representing white noise inputs, and $A(t)$ and $B(t)$ are time-varying system matrices. As shown in [6-12], the state covariance matrix $P(t)$, defined by $P(t) \triangleq E\{\underline{x}(t)\underline{x}^T(t)\}$, satisfies the matrix differential equation

$$\dot{P} = AP + P^TA^T + BQB^T \qquad (2)$$

where the functional dependence on time t is implied throughout. The input covariance matrix Q is defined by the relationship

$$E\{\underline{w}(t)\underline{w}^T(\tau)\} = Q(t)\delta(t-\tau) \qquad (3)$$

Let a large-scale system described by (1) be segmented into several subsystems as shown in Figure 1. In the subsystem context, the matrices A, B, P, and Q may be partitioned as

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1N} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ A_{N1} & \cdots & A_{NN} \end{pmatrix} \qquad B = \begin{pmatrix} B_{11} & \cdots & B_{1N} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ B_{N1} & \cdots & B_{NN} \end{pmatrix}$$

SWIEEECO Record of Technical Papers, Dallas, Texas, April 19-21, 1972, pp. 135-138.

$$P = \begin{pmatrix} P_{11} & \cdots & P_{1N} \\ \vdots & & \vdots \\ P_{N1} & \cdots & P_{NN} \end{pmatrix} \qquad Q = \begin{pmatrix} Q_{11} & \cdots & Q_{1N} \\ \vdots & & \vdots \\ Q_{N1} & \cdots & Q_{NN} \end{pmatrix} \tag{4}$$

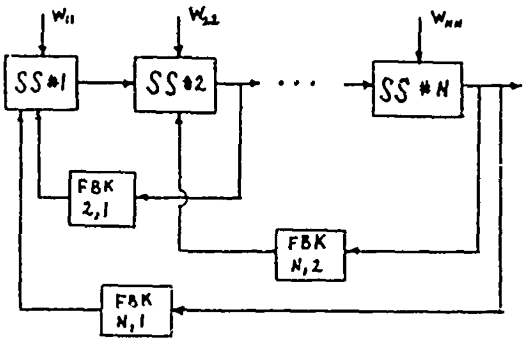


Figure 1. Schematic Diagram of a Large-Scale System Showing Individual Subsystem Connections.

To simplify the development which follows, it is assumed that noise inputs are uncorrelated with each other and, furthermore, that each enters only a single designated subsystem as shown in Figure 1. This assumption means that both B and Q are diagonal matrices.

Since the matrix P is symmetric, i.e. $P = P^T$, one has $P_{ij} = P^T_{ji}$. Therefore, (2) may be expressed as

$$\begin{aligned} \dot{P}_{ij} &= A_{i1}P_{1j} + A_{i2}P_{2j} + \cdots A_{iN}P_{Nj} \\ &\quad + P^T_{1i}A^T_{j1} + P^T_{2i}A^T_{j2} + \cdots P^T_{Ni}A^T_{jN} \\ &\quad + B_{ii}Q_{ii}B^T_{ii}\delta_{ij} \end{aligned} \tag{5}$$

where δ_{ij} is zero if $i \neq j$ and unity if $i = j$. Equation (5) may be conveniently grouped as

$$\begin{aligned} \dot{P}_{ij} &= \sum_{k=1}^{i-1} A_{ik}P_{kj} + A_{ii}P_{ij} \\ &\quad + \sum_{k=i+1}^{N} A_{ik}P_{kj} + \sum_{k=1}^{j-1} P^T_{ki}A^T_{jk} + P^T_{ji}A^T_{jj} \\ &\quad + \sum_{k=j+1}^{N} P^T_{ki}A^T_{jk} + B_{ii}Q_{ii}B^T_{ii}\delta_{ij} \end{aligned} \tag{6}$$

for $i = 1,\ldots N$ and $j = 1,\ldots,N$.

The matrices P^T_{ki} and P^T_{ji} in the second line of (6) may be replaced by P_{ik} and P_{ij}, respectively, since P is symmetric. The second and fifth terms in (6) are the only ones involving P_{ij}. Moreover, the first and fourth terms have as coefficient matrices entries from the lower left of the main diagonal of the system matrix A. These terms represent feedforward paths. Elements from the upper right of the main diagonal of A appear in the third and sixth terms in (6). These represent feedback paths. The seventh, or last, term represents input noise data. Rewriting (6) and summarizing these observations, one has

$$\begin{aligned} \dot{P}_{ij} &= \underbrace{A_{ii}P_{ij} + P_{ij}A^T_{jj}}_{\text{Self-Interaction}} \\ &\quad + \underbrace{\sum_{k=1}^{i-1} A_{ik}P_{kj} + \sum_{k=1}^{j-1} P_{ik}A^T_{jk}}_{\text{Feed-Forward Paths}} \\ &\quad + \underbrace{\sum_{k=i+1}^{N} A_{ik}P_{kj} + \sum_{k=j+1}^{N} P_{ik}A^T_{jk}}_{\text{Feedback Paths}} \\ &\quad + \underbrace{B_{ii}Q_{ii}B^T_{ii}\delta_{ij}}_{\text{Noisy Inputs}} \end{aligned} \tag{7}$$

What is desired is to apply (7) sequentially to determine P_{ij} for all i and all j. It is particularly important only to know P_{ii}, but it may be shown that calculation for all i and j is necessary to completely determine P_{ii}.

A flow chart showing details of the sequential algorithm for covariance matrix calculations is given in Figure 2. Numerical results obtained by using a computer software package developed from Figure 2 are provided in a later section of this paper.

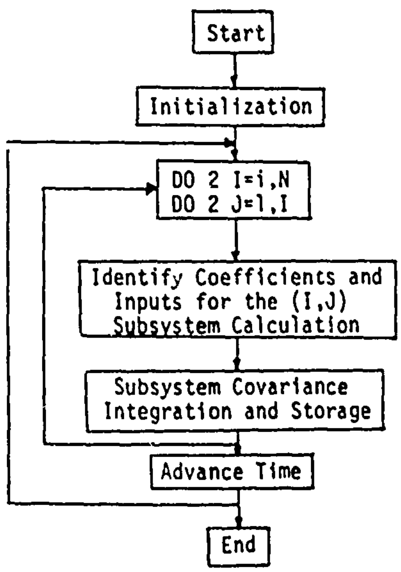


Figure 2. A Flow Chart of the Sequential Algorithm.

3. ERROR CONSIDERATIONS

The use of sequential calculations makes available current values of subsystem covariance matrices only for feedforward terms in subsequent equations. Previous values must be used as approximations in feedback terms, which is equivalent to having samplers and zero-order hold devices in certain feedback loops in covariance matrix calculations.

To illustrate the nature of these approximations consider the system of Figure 3, which has a single feedback loop around N cascaded subsystems. Many of the elements of the A matrix are zero for this given system structure. In fact, except for the element A_{1N}, which is due to the single feedback loop, only the main diagonal elements and those immediately below and adjacent to the main diagonal are non-zero. Figure 4 shows a block diagram for the state covariance matrix elements and indicates that the single feedback loop in Figure 3 introduces a feedback loop for every row of the covariance matrix. Let N=5 for a particular system.

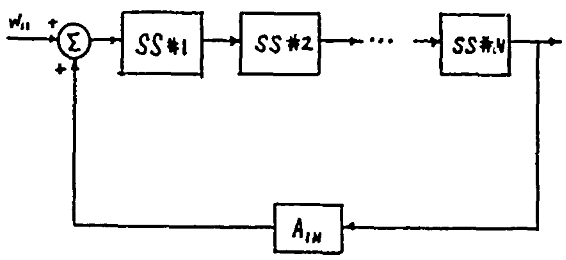


Figure 3. A Simple Feedback System.

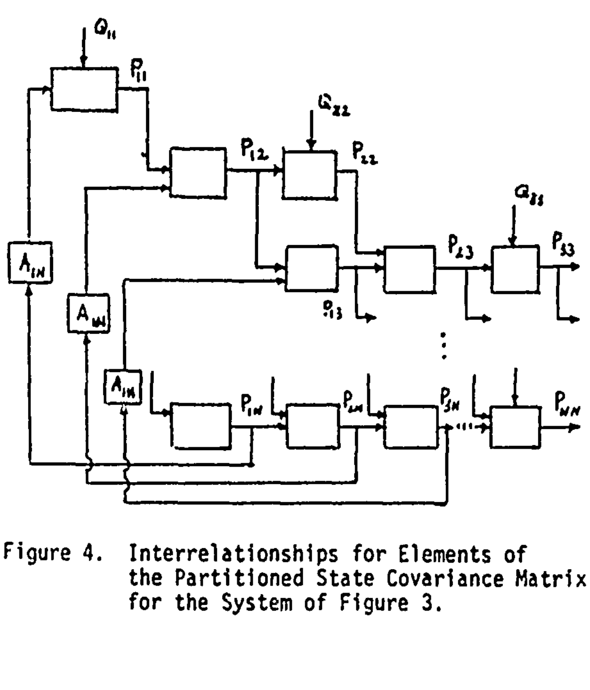


Figure 4. Interrelationships for Elements of the Partitioned State Covariance Matrix for the System of Figure 3.

For the jth row, the previous value of P_{jN} is needed. For example, to calculate P_{44} one may write

$$\dot{P}_{44} = A_{44}P_{44} + P_{44}A_{44}^T + A_{43}P_{34} + P_{34}A_{43}^T + B_{44}Q_{44}B_{44}^T$$

$$\dot{P}_{34} = A_{33}P_{34} + P_{34}A_{44}^T + A_{32}P_{24} + P_{33}A_{43}^T$$

$$\dot{P}_{24} = A_{22}P_{24} + P_{24}A_{44}^T + A_{21}P_{14} + P_{23}A_{43}^T$$

$$\dot{P}_{14} = A_{11}P_{14} + P_{14}A_{44}^T + P_{13}A_{43}^T + A_{15}P_{45} \qquad (8)$$

In interpreting (8), one should note that the four matrix equations must be applied sequentially in reverse order, beginning with the last. The matrices P_{13}, P_{23}, and P_{33} are known from calculations for the previous subsystem, i.e. subsystem #3. Observe that the last equation in (8) has the term $A_{15}P_{45}$, which is obtained from computations at the previous time interval and used as an approximation for the current interval.

This section has shown the kind of approximation needed for applying the sequential algorithm. The next section describes the reduction in computer storage for the algorithm.

4. REDUCTION IN COMPUTER STORAGE

There is a certain amount of digital computer core required for simply storing the matrices A, B, Q, and P. With some important exceptions, these matrices are needed regardless of the method being used to solve the matrix differential equation (2). Of major concern here is the comparison of additional dimensioned core locations required by the sequential algorithm and by the direct evaluation of (2) using standard numerical integration formulas.

Consider the case of m cascaded subsystems with each of order r. Euler's Method would require $(mr)^2$ additional locations, i.e. for the P matrix, by direct evaluation. However, only $2\,m^2 + 2\,r^2$ additional locations are needed for the sequential algorithm. If m is large and also much greater than r, then the savings in core can be significant. Moreover, the additional core for RK2, i.e. the standard second-order Runge-Kutta formula, is $3(mr)^2$ by direct evaluation and $2\,m^2 + 4\,r^2$ by the sequential algorithm. Corresponding core requirements for RK4 are $3(mr)^2$ and $3\,m^2 + 4\,r^2$, respectively. Figure 5 shows plots of ρ versus m, where ρ is the ratio of additional core required by direct evaluation to the additional core required by the sequential algorithm. These curves should be viewed as rough estimates, rather than exact ratios, since use of symmetry conditions and other more efficient programming techniques would alter these curves somewhat. It should be pointed out that if a large amount of core is required for the system matrices and for program operations, then a dramatic per cent reduction in the additional core required by the sequential algorithm may be de-emphasized when considered on the basis of total core requirements.

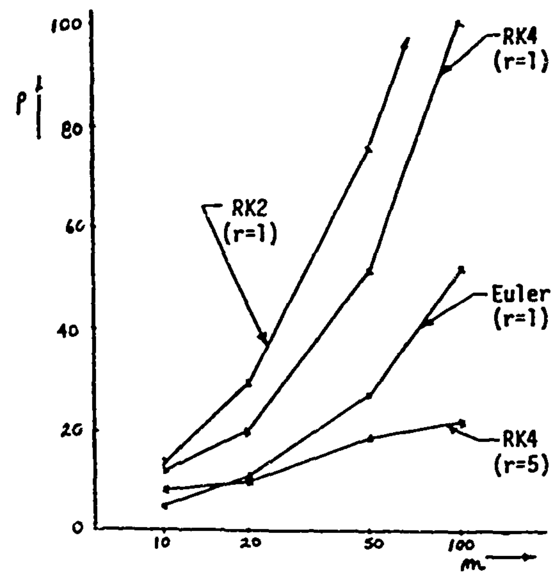


Figure 5. Ratio (ρ) of Additional Core Needed by Direct Evaluation to Sequential Algorithm Versus Number (m) of Subsystems.

5. AN EXAMPLE

The sequential algorithm was compared with the direct evaluation method on an open-loop system consisting of several cascaded first-order subsystems. The direct method used RK2 for numerical integration, and the sequential algorithm used RK4. It was verified first that the two methods gave essentially identical numerical results for ten subsystems. Moreover, these results agreed with 100 Monte Carlo simulation runs performed on the same tenth-order system. The procedure for comparing total core requirements was to increase the number of subsystems considered by each method until a preset level of 100K bytes of core was exceeded. It was found that the direct evaluation method could handle only about 45 cascaded subsystems. However, the sequential algorithm could be used for as many as 130 subsystems.

6. CONCLUSIONS

A sequential algorithm has been developed for reducing digital computer core requirements in covariance matrix calculations. The associated computations are performed on a subsystem basis, and integration variables are re-used from subsystem to subsystem. A particular example has demonstrated that a significant savings in core requirements can be realized by the new algorithm.

7. ACKNOWLEDGEMENTS

The research reported here was initiated during the summer of 1971 in the Guidance and Control Directorate of the Research, Development, Engineering and Missile Systems Laboratory, U. S. Army Missile Command, Redstone Arsenal, Alabama, under the ARO/D Army Laboratory Research Cooperative Program, Contract DAHC04-68-C-0011. Later work was performed both at the U. S. Army Missile Command and within the School of Electrical Engineering at Oklahoma State University. Research support is acknowledged from Institutional Research Funds and from the Center for Systems Science at Oklahoma State University. The authors wish to express their appreciation to Mr. V. M. Gupta for his assistance in making computer runs for the example on the IBM 360/65 at Oklahoma State University.

REFERENCES

(1) James R. Rowland and Willard M. Holmes, "Stochastic Analysis Techniques for Error Propagation in Large-Scale Missile Systems," Technical Report No. RG-TR-71-19, Guidance and Control Directorate, U. S. Army Missile Command, Redstone Arsenal, Alabama, August , 1971.

(2) Willard M. Holmes," Hybrid Simulation of Large Scale Missile Systems with Medium - Sized Hybrid Facilities," Proceedings of the Third Annual Southeastern Symposium on System Theory, Vol. 1, Georgia Institute of Technology, Atlanta, Ga., April 5-6, 1971.

(3) Mihajlo D. Mesarovic, "Multilevel Systems and Concepts in Process Control," Proceedings of the IEEE, Vol. 58, No. 1, pp. 111-125, January, 1970.

(4) Irving Lefkowitz, "An Approach to Computer Control Research in a University Environment," Proceedings of the IEEE, Vol. 58, No. 1, pp. 125-132, January, 1970.

(5) A. R. M. Noton, "Two Level Form of the Kalman Filter," IEEE Transactions on Automatic Control, Vol. AC-16, No. 2, pp. 128-133, April, 1971.

(6) R. E. Kalman and R. S. Bucy, " New Results in Linear Filtering and Prediction Theory," ASME Transactions: Journal of Basic Engineering, Vol. 83D, pp. 95-108, March, 1961.

(7) R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," ASME Transactions: Journal of Basic Engineering, Vol. 82D, pp. 35-45, March 1960.

(8) Andrew P. Sage, Optimum Systems Control, Prentice-Hall, 1968, ... 3.2, pp. 220-226.

(9) Andrew P. Sage ... M. Melsa, Estimation Theory: With Ap...tions to Communications and Control, Mc...-Hill, 1970.

(10) James S. Meditch, Stochastic Optimal Linear Estimation and Control, McGraw-Hill, 1969, Sec. 4.3 and 4.4, pp. 125-152.

(11) Paul B. Liebelt, An Introduction to Optimal Estimation, Addison-Wesley, 1967, Sec. 4.10 and 4.16, pp. 112-134.

(12) Arthur E. Bryson, Jr. and Yu-Chi Ho, Applied Optimal Control, Blaisdell Publishing Company, Sec. 11.4 and 11.5, pp. 328-344, 1969.

(13) Andrew H. Jazwinski, Stochastic Processes and Filtering Theory, Academic Press, 1970.

(14) James S. Meditch, "A Class of Suboptimal Linear Controls," IEEE Transactions on Automatic Control, Vol. AC-11, pp. 433-439, July, 1966.

Comput. & Elect. Engng. Vol. 1, pp. 111-118. Pergamon Press, 1973. Printed in Great Britain

OPTIMAL DIGITAL SIMULATIONS FOR RANDOM LINEAR SYSTEMS WITH INTEGRATION CONSTRAINTS

JAMES R. ROWLAND

School of Electrical Engineering and Center for Systems Science, Oklahoma State University, Stillwater, Oklahoma 74074, U.S.A.

(*Received 20 November 1972*)

Abstract—A generalized approach involving concepts from optimization theory is developed for realizing optimal digital simulations for linear, time-varying, continuous dynamical systems having random inputs by modifying discrete input signal variances. The minimization of a cost functional based on the state covariance matrices of the continuous system and its discrete model leads to a two-point boundary value problem which can be solved by known numerical techniques. The result is a systematic procedure for determining optimal digital simulations under the constraints that the numerical integration formula and integration step size have been specified in advance. An example is presented to illustrate the procedure, including a verification using Monte Carlo simulation runs.

INTRODUCTION

Increased digital and hybrid computer capabilities in recent years have resulted in an even stronger reliance on the Monte Carlo approach for statistical analyses of large-scale dynamical systems[1, 2]. Improved digital random number generators[3, 4] have already been developed for producing more precise statistical inputs. Emphasis has also been placed on developing more accurate[5], as well as more efficient[6, 7], numerical algorithms for digitally integrating large systems of continuous differential equations. Moreover, the rapidly expanding field of digital signal processing has only recently opened up several new possibilities for handling continuous systems efficiently via digital representations [8–10]. Many of the previous works, e.g. [8] and [11], are based on matching the frequency spectra of continuous systems and discretized models. Although more extensive time-domain techniques have been reported in the literature[12], only the simple rectangular, or Euler, approximation is in common use for discretizing continuous systems[13–16].

This paper utilizes concepts from optimization theory to derive a time-domain solution to the problem of determining optimal digital representations for random linear continuous systems having integration constraints. The stochastic formulation reduces to a deterministic two-point boundary value problem in the calculus of variations, which can be solved by known techniques. Such integration constraints can occur, for example, when large-scale systems are being simulated on medium-sized hybrid facilities[17]. If the analog equipment is seriously limited, then a few of the integrations must be performed digitally. In these cases, the integration method and corresponding step size are often constrained quite severely. The purpose of this paper is to present a systematic procedure for modifying the digital simulation input signals to yield optimal results.

PROBLEM FORMULATION

Consider a continuous, linear, time-varying system described by

$$\dot{x}(t) = A(t)x(t) + B(t)w_c(t) \tag{1}$$

where $x(t)$ is an n-vector representing the system state, $w_c(t)$ is an m-vector of white noise input disturbances, and $A(t)$ and $B(t)$ are n by n and n by m system matrices, respectively. The white noise input $w_c(t)$ has a mean of zero and a covariance matrix $Q_c(t)$ defined by

$$E\{w_c(t)w_c^T(\tau)\} = Q_c(t)\delta_d(t-\tau) \tag{2}$$

where $\delta_d(\cdot)$ is the Dirac delta function. Let a discrete model of the continuous system (1) have the form

$$y(t_{k+1}) = \Phi_d(t_{k+1}, t_k)y(t_k) + H_d(t_{k+1}, t_k)w_d(t_k) \tag{3}$$

where $y(t_k)$ is an n-vector representing the model state at time $t_k = kT$, $w_d(t_k)$ is an m-vector input sequence of random numbers, and Φ_d and H_d are n by n and n by m time-varying model matrices, respectively. The zero-mean model input $w_d(t_k)$ has a covariance matrix $Q_d(t_k)$ given by

$$E\{w_d(t_k)w_d^T(t_j)\} = \begin{cases} Q_d(t_k) & \text{for } k = j \\ 0 & \text{for } k \neq j. \end{cases} \tag{4}$$

The cost functional is

$$J[Q_d(t_k)] = \text{Trace} \sum_{k=0}^{K-1} \tfrac{1}{2}[P_c(t_{k+1}) - P_d(t_{k+1})]^T R[P_c(t_{k+1}) - P_d(t_{k+1})] \tag{5}$$

where R is some positive semidefinite n by n matrix and $P_c(t)$ and $P_d(t_k)$ are defined by

$$P_c(t) \triangleq E\{x(t)x^T(t)\} \tag{6}$$

$$P_d(t_k) \triangleq E\{y(t_k)y^T(t_k)\}. \tag{7}$$

The problem is to determine $Q_d(t_k)$ such that the cost functional J in (5) is minimized for specified model matrices Φ_d and H_d in (3) corresponding to a given numerical integration formula and integration step size T.

DEVELOPMENT OF OPTIMAL DIGITAL SIMULATIONS

The approach to be utilized here is to determine the matrix difference equation for $P_d(t_{k+1})$ in terms of $P_d(t_k)$ and the input covariance matrix $Q_d(t_k)$. Thereafter, the cost functional in (5) may be minimized with respect to $Q_d(t_k)$ by invoking known results from optimization theory.

Using the model difference equation (3) in the definition (7) yields

$$\begin{aligned} P_d(t_{k+1}) &= E\{y(t_{k+1})y^T(t_{k+1})\} \\ &= E\{[\Phi_d y(t_k) + H_d w_d(t_k)][\Phi_d y(t_k) + H_d w_d(t_k)]^T\} \end{aligned}$$

$$P_d(t_{k+1}) = \Phi_d(t_{k+1}, t_k)P_d(t_k)\Phi_d^T(t_{k+1}, t_k) + H_d(t_{k+1}, t_k)Q_d(t_k)H_d^T(t_{k+1}, t_k). \tag{8}$$

Therefore, the optimal digital simulation problem originally stated has been reduced to a two-point boundary value problem in the calculus of variations. It is required to minimize the cost functional (5) subject to the matrix difference equation constraint given by (8).

Before proceeding with this optimization solution, it is instructive for comparison purposes to determine the corresponding difference equation for $P_c(t_{k+1})$. The exact expression for $P_c(t)$ may be obtained, as shown in [13], by solving for $\mathbf{x}(t)$ from (1) and substituting the result into the defining equation (6), i.e.

$$\begin{aligned} P_c(t) &= E\{\mathbf{x}(t)\mathbf{x}^T(t)\} \\ &= E\left\{\left[\Phi_c(t,t_0)\mathbf{x}(t_0) + \int_{t_0}^{t}\Phi_c(t,\tau)B(\tau)\mathbf{w}_c(\tau)\mathrm{d}\tau\right]\right. \\ &\quad \left.\cdot\left[\Phi_c(t,t_0)\mathbf{x}(t_0) + \int_{t_0}^{t}\Phi_c(t,\tau)B(\tau)\mathbf{w}_c(\tau)\mathrm{d}\tau\right]^T\right\}. \end{aligned} \tag{9}$$

Performing the indicated multiplications in (9) and noting that $\mathbf{x}(t_0)$ and $\mathbf{w}_c(t)$ are uncorrelated, one has

$$\begin{aligned} P_c(t) &= \Phi_c(t,t_0)E\{\mathbf{x}(t_0)\mathbf{x}^T(t_0)\}\Phi_c^T(t,t_0) \\ &\quad + \int_{t_0}^{t}\int_{t_0}^{t}\Phi_c(t,\tau)B(\tau)E\{\mathbf{w}_c(\tau)\mathbf{w}_c^T(\rho)\}B^T(\rho)\Phi^T(t,\rho)\mathrm{d}\tau\mathrm{d}\rho. \end{aligned} \tag{10}$$

Using (2) and the sifting property of the delta function gives, for $t = t_{k+1}$ and $t_0 = t_k$, the recursive relationship

$$\begin{aligned} P_c(t_{k+1}) &= \Phi_c(t_{k+1},t_k)P_c(t_k)\Phi_c^T(t_{k+1},t_k) \\ &\quad + \int_{t_k}^{t_{k+1}}\Phi_c(t_{k+1},\tau)B(\tau)Q_c(\tau)B^T(\tau)\Phi_c^T(t_{k+1},\tau)\mathrm{d}\tau. \end{aligned} \tag{11}$$

The matrix equation in (11) is not a constraint equation for the posed optimization problem because $P_c(t_{k+1})$ is not a function of the optimization variables contained in the matrix $Q_d(t_k)$. On the contrary, $P_c(t_{k+1})$ is simply treated in (5) as some known time-varying matrix which is to be modeled by $P_d(t_{k+1})$.

It is known from the calculus of variations in optimization theory that the solution to the posed problem requires the introduction of an n by n matrix $\lambda_d(t_k)$ of Lagrange multipliers for $P_d(t_k)$. Moreover, $\lambda_d(t_k)$ satisfies a matrix adjoint difference equation which has the boundary condition

$$\lambda_d(t_K) = 0 \tag{12}$$

where t_K is that terminal time indicated in (5). A convenient method for obtaining the adjoint equation is to define the Hamiltonian H as

$$H = \mathrm{Trace}\{\tfrac{1}{2}[P_c(t_{k+1}) - P_c(t_{k+1})]^T R[P_c(t_{k+1}) - P_d(t_{k+1})] + P_d(t_{k+1})\lambda_d^T(t_{k+1})\}. \tag{13}$$

It has been shown[13] that the matrix adjoint equation is

$$\lambda_d(t_k) = \frac{\partial H}{\partial P_d(t_k)}. \tag{14}$$

Equation (8) must be substituted into (13) before the indicated partial differentiation in (14) is performed. Finally, the optimal value of $Q_d(t_k)$ satisfies

$$\frac{\partial H}{\partial Q_d(t_k)} = 0. \tag{15}$$

The resulting two-point boundary problem for determining the optimal digital simulation involves solving simultaneously the equations in (8), (14) and (15) with the boundary conditions in (12) for $\lambda_d(t_k)$ and known initial conditions for $P_d(t_k)$, i.e. $P_d(t_0) = P_{d0}$.

It should be observed that a degenerate case of the optimization problem occurs when the number (m) of random inputs is at least as large as the number (n) of system states, i.e. $m \geq n$. In such a case, the cost functional in (5) becomes zero. If, in addition, the model is permitted to utilize the state transition matrix (STM) method of integration[7], then $Q_d(t_k)$ becomes $Q_c(t_k)/T$, as shown in Ref. 18.

OPTIMAL NUMERICAL RESULTS

As a particular example for purposes of numerical comparisons, consider the second-order system given by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -2x_1 - 3x_2 + w_c(t) \end{aligned} \tag{16}$$

where $x_1(0) = x_2(0) = 0$ and $w_c(t)$ is a zero-mean white noise process with $Q_c \equiv 1$. Let the discrete model matrices $\Phi_d(t_{k+1}, t_k)$ and $H_d(t_{k+1}, t_k)$ in (3) be considered for two separate cases corresponding to the use of the Euler and second-order Runge–Kutta (RK2) integration formulas on (16). Since (16) is a linear time-invariant system, Φ_d and H_d are functions only of the integration step size T, where $T = t_{k+1} - t_k$. For Euler's formula, these matrices are

$$\begin{aligned} \Phi_d(T) &= \begin{pmatrix} \phi_{11}(T) & \phi_{12}(T) \\ \phi_{21}(T) & \phi_{22}(T) \end{pmatrix} = \begin{pmatrix} 1 & T \\ -2T & 1-3T \end{pmatrix} \\ H_d(T) &= \begin{pmatrix} h_1(T) \\ h_2(T) \end{pmatrix} = \begin{pmatrix} 0 \\ T \end{pmatrix} \end{aligned} \tag{17}$$

and for the RK2 formula

$$\begin{aligned} \Phi_d(T) &= \begin{pmatrix} 1 - T^2 & T - 1{\cdot}5T^2 \\ -2T + 3T^2 & 1 - 3T + 3{\cdot}5T^2 \end{pmatrix} \\ H_d(T) &= \begin{pmatrix} 0{\cdot}5T^2 \\ T - 1{\cdot}5T^2 \end{pmatrix}. \end{aligned} \tag{18}$$

Let the cost functional J in (5) be defined by

$$J = \frac{1}{2}\sum_{k=0}^{K-1} \{[p_{c11}(t_{k+1}) - p_{d11}(t_{k+1})]^2 + [p_{c22}(t_{k+1}) - p_{d22}(t_{k+1})]^2\} \tag{19}$$

where K is selected in various parts of this problem such that the product KT is approximately 5 sec.

The component equations for P_d corresponding to (8) are

$$\begin{aligned} p_{d11}(t_{k+1}) &= \phi_{11}^2 p_{d11}(t_k) + 2\phi_{11}\phi_{12}p_{d12}(t_k) + \phi_{12}^2 p_{d22}(t_k) + h_1^2 Q_d(t_k) \\ p_{d12}(t_{k+1}) &= \phi_{11}\phi_{21}p_{d11}(t_k) + (\phi_{11}\phi_{22} + \phi_{12}\phi_{21})p_{d12}(t_k) \\ &\quad + \phi_{12}\phi_{22}p_{d22}(t_k) + h_1 h_2 Q_d(t_k) \\ p_{d22}(t_{k+1}) &= \phi_{21}^2 p_{d11}(t_k) + 2\phi_{21}\phi_{22}p_{d12}(t_k) + \phi_{22}^2 p_{d22}(t_k) + h_2^2 Q_d(t_k) \end{aligned} \tag{20}$$

with each having zero initial conditions. The Hamiltonian in (13) is

$$H = \tfrac{1}{2}[Y_{11}]^2 + \tfrac{1}{2}[Y_{22}]^2 + [\phi_{11}^2 p_{d11}(t_k) + 2\phi_{11}\phi_{12}p_{d12}(t_k) + \phi_{12}p_{d22}(t_k) + h_1^2 Q_d(t_k)]\lambda_{d11}(t_{k+1}) + [\phi_{11}\phi_{21}p_{d11}(t_k) + (\phi_{11}\phi_{22} + \phi_{12}\phi_{21})p_{d12}(t_k) + \phi_{12}\phi_{22}p_{d22}(t_k) + h_1 h_2 Q_d(t_k)]\lambda_{d12}(t_{k+1}) + [\phi_{21}^2 p_{d11}(t_k) + 2\phi_{21}\phi_{22}p_{d12}(t_k) + \phi_{22}^2 p_{d22}(t_k) + h_2^2 Q_d(t_k)]\lambda_{d22}(t_k) \tag{21}$$

where Y_{11} and Y_{22} are defined as

$$\begin{aligned} Y_{11} &= p_{c11}(t_{k+1}) - \phi_{11}^2 p_{d11}(t_k) - 2\phi_{11}\phi_{12}p_{d12}(t_k) - \phi_{12}^2 p_{d22}(t_k) - h_1^2 Q_d(t_k) \\ Y_{22} &= p_{c22}(t_{k+1}) - \phi_{21}^2 p_{d11}(t_k) - 2\phi_{21}\phi_{22}p_{d12}(t_k) - \phi_{22}^2 p_{d22}(t_k) - h_2^2 Q_d(t_k). \end{aligned} \tag{22}$$

Moreover, the component equations for the adjoint matrix λ_d in (14) are

$$\begin{aligned} \lambda_{d11}(t_k) &= \frac{\partial H}{\partial p_{d11}(t_k)} = -\phi_{11}^2[Y_{11}] - \phi_{21}^2[Y_{22}] + \phi_{11}^2\lambda_{d11}(t_{k+1}) \\ &\qquad + \phi_{12}\phi_{21}\lambda_{d12}(t_{k+1}) + \phi_{21}^2\lambda_{d22}(t_{k+1}) \\ \lambda_{d12}(t_k) &= \frac{\partial H}{\partial p_{d12}(t_k)} = -2\phi_{11}\phi_{12}[Y_{11}] - 2\phi_{21}\phi_{22}[Y_{22}] + 2\phi_{11}\phi_{12}\lambda_{d11}(t_{k+1}) \\ &\qquad + (\phi_{11}\phi_{22} + \phi_{12}\phi_{21})\lambda_{d12}(t_{k+1}) + 2\phi_{21}\phi_{22}\lambda_{d22}(t_{k+1}) \\ \lambda_{d22}(t_k) &= \frac{\partial H}{\partial p_{d22}(t_k)} = -\phi_{12}^2[Y_{11}] - \phi_{22}^2[Y_{22}] + \phi_{12}^2\lambda_{d11}(t_{k+1}) \\ &\qquad + \phi_{12}\phi_{22}\lambda_{d12}(t_{k+1}) + \phi_{22}^2\lambda_{d22}(t_{k+1}). \end{aligned} \tag{23}$$

The standard formulation for the two-point boundary value problem requires the inversion of the three equations in (23) to yield $\lambda_d(t_{k+1})$ in terms of $\lambda_d(t_k)$ and $P_d(t_k)$. Using (15) then gives $Q_d(t_k)$ as a function of $\lambda_d(t_{k+1})$ and $P_d(t_k)$, which can further be written in terms of P_d and λ_d at time t_k. However, the split boundary conditions at t_0 and t_K makes an approximate iterative solution, such as the gradient technique, highly desirable.

One version of the gradient technique[13] utilizes the equations in (20) and (23) directly without inverting (23) or solving (15) for $Q_d(t_k)$. Letting $Q_d(t_k) = Q_c/T$ for the first iteration, the P_d component equations in (20) were solved forward in time. Thereafter, (23) was solved backwards in time using the boundary conditions in (12). The value of $Q_d(t_k)$ for the next iteration was obtained by adding to the previous value the term $-\alpha[\partial H/\partial Q_d(t_k)]$, which had been evaluated for the P_d and λ_d of the last iteration. For this example, a proportionality constant α of 200 to 500 resulted in the convergence of this repetitive process in ten iterations or less for most of the cases considered. The average optimal values of Q_d obtained by this gradient procedure are presented in Table 1, since the optimal $Q_d(t_k)$

Table 1. Optimal discrete model input variances Q_d for several cases

Numerical integration formula	Step size (T)	Number of steps (K)	Average optimal $Q_d(t_k)$
Euler	0·1	50	8·1
Euler	0·2	25	3·3
RK2	0·2	25	5·6
RK2	0·3	17	4·3

for these cases were within 10 per cent of these averages for all t_k. Observe that the optimal Q_d for these constrained discrete model cases varied considerably from the unconstrained model solutions ($Q_d = Q_c/T$). For example, the average optimal Q_d for Euler's method with $T = 0{\cdot}1$ was 8·1, while Q_d for the unconstrained problem was 10. Figure 1 shows the cost functional J for the tabulated cases as a function of t_k. These curves verify the expected result that a larger J is obtained when the discrete model utilizes a less accurate integration formula and a larger step size.

Variances for both x_1 and x_2 are plotted as functions of time in Fig. 2 for Euler's method with $T = 0{\cdot}1$. Nonoptimal solutions obtained by arbitrarily selecting $Q_d = Q_c/T = 10$ show good agreement between p_{d11} and p_{c11} but extremely poor results for representing p_{c22} by p_{d22}. On the other hand, corresponding curves obtained by using the optimal

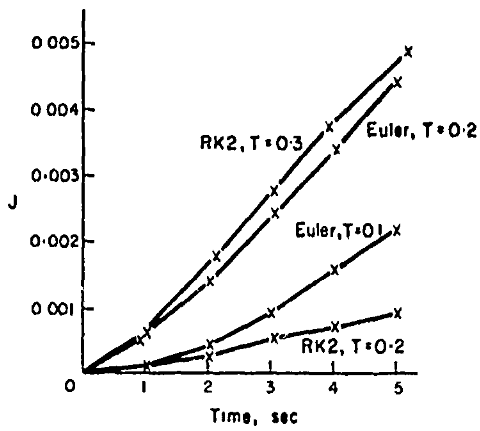


Fig. 1. Plots of J vs time.

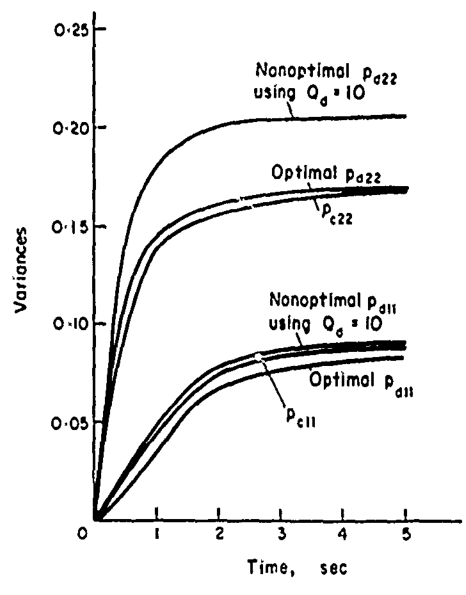


Fig. 2. A comparison of optimal and nonoptimal solutions for Euler's method with $T = 0{\cdot}1$.

values of $Q_d(t_k)$ distribute the error more evenly between the two main diagonal components of P_d, which is necessary to minimize the cost functional in (19). Monte Carlo simulation runs were ensemble-averaged on the digital computer to verify these optimization results. Figure 3 shows that 100 Monte Carlo runs were insufficient for both the constrained discrete model using Euler's method with $T = 0{\cdot}1$ and the unconstrained problem using a more accurate integration formula and smaller step size. For this example, Monte Carlo runs for the unconstrained problem utilized RK2 with $T = 0{\cdot}05$, which yielded a negligibly small cost functional ($J \cong 0$) in the gradient optimization procedure. Following the guidelines specified in Refs. 19 and 20, it was found that 1000 Monte Carlo runs gave results which agreed quite well with the variances determined in Fig. 2.

EXTENSIONS

The optimal digital simulation techniques developed in this paper for specified Φ_d and H_d can easily be extended to permit these discrete model matrices to have free optimization parameters. The resulting formulation would require the optimal selection of both the discrete input covariance matrix $Q_d(t_k)$ and certain discrete model parameters in Φ_d and H_d. This additional flexibility in the optimization procedure would result in a reduction in the cost functional by an amount depending upon precisely how these model parameters affect the dynamical system response. A special case of this formulation has been considered in [18].

The extension of these optimization results to mildly nonlinear systems can be achieved by utilizing linearized variational equations about a nominal solution. As shown in [20], the application of the error propagation algorithm in equation (11) for an approximate analysis of low-noise, mildly nonlinear systems has yielded acceptable results. Further digital simulation improvements might be realized by simultaneously optimizing the nominal solution and the discrete linearized variational model[21]. Finally, it appears that the concepts developed here for optimal digital simulations might also be extended for

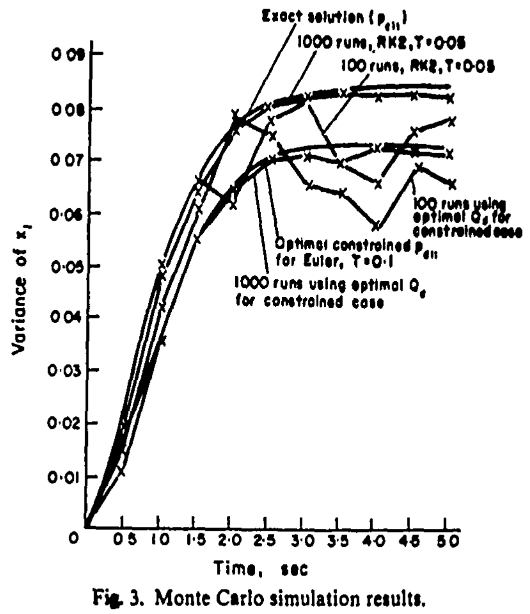


Fig. 3. Monte Carlo simulation results.

the optimal discrete implementation of stochastic filtering algorithms in continuous dynamical systems[22].

CONCLUSIONS

Known optimization techniques have been applied to obtain optimal digital simulations for random linear systems having integration constraints. The developed procedure depends upon optimally selecting the input covariance matrix $Q_d(t_k)$ for prespecified discrete model matrices corresponding to fixed numerical integration formulas with a given step size. An example including Monte Carlo simulation runs has been presented to demonstrate the improvements over arbitrary nonoptimal solutions.

REFERENCES

1. G. A. Bekey and W. J. Karplus, *Hybrid Computation*. Wiley, New York (1968).
2. A. Ralston, *A First Course in Numerical Analysis*. McGraw-Hill, New York (1965).
3. R. P. Chambers, Random number generation on digital computers, *IEEE Spectrum*, Vol. 4, No. 2, pp. 48–56 (1967).
4. R. J. Brown, Jr., and J. R. Rowland, Autocorrelation significance in digital pseudo-random number generation, Tech. Rep., School of Electrical Engineering, Georgia Institute of Technology, Atlanta, Georgia, pp. 1–20, March 1970.
5. F. Scheid, *Numerical Analysis*, Schaum's Outline Series, pp. 202–204. McGraw-Hill, New York (1968).
6. P. R. Benyon, A review of numerical methods for digital simulation, *Simulation*, 11, 219–238 (1968).
7. J. R. Rowland and W. M. Holmes, A variational approach to digital integration, *IEEE Transactions on Computers*, Vol. C-20, No. 8, pp. 894–900 (1971).
8. B. Gold and C. M. Rader, *Digital Processing of Signals*. McGraw-Hill, New York (1969).
9. J. F. Kaiser, Digital filters, Chapter 7 in *Systems Analysis by Digital Computer* (Edited by F. F. Kuo and J. F. Kaiser), pp. 218–285. Wiley, New York (1966).
10. J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures*. Interscience, New York (1971).
11. L. M. McCloskey, Jr., J. C. Eck and R. E. Medeiros, Generation of stochastic processes for digital computer simulations, Tech. Memo. No. 6, Deep Submergence Systems: Navigation, Test, and Analysis Group; M. I. T. Instrumentation Laboratory (1968).
12. P. D. Krut'ko, *Statistical Dynamics of Sampled Data Systems* (Translated from Russian by Scripta Technica Ltd.). Iliffe, London (1969).
13. A. P. Sage, *Optimum Systems Control*. Prentice-Hall, Englewood Cliffs, N.J. (1968).
14. A. P. Sage and J. L. Melsa, *Estimation Theory with Applications to Communications and Control*. McGraw-Hill, New York (1971).
15. A. E. Bryson, Jr. and Yu-Chi Ho, *Applied Optimal Control*. Blaisdell, Waltham, Mass. (1969).
16. J. S. Meditch, *Stochastic Optimal Linear Estimation and Control*. McGraw-Hill, New York (1969).
17. W. M. Holmes, Hybrid simulation of large scale missile systems with medium-sized hybrid facilities, *Proceedings of the Third Annual Southeastern Symposium on System Theory*, Vol. 1, Georgia Institute of Technology, Atlanta, Ga., Paper F1, 5–6 April (1971).
18. J. R. Rowland and V. M. Gupta, Digital simulations for Monte Carlo analysis, *Proc. 15th Midwest Symp. on Circuit Theory*, Vol. 1, Paper V.3, University of Missouri-Rolla, 4–5 May (1972).
19. R. S. Bucy, Realization of nonlinear filters, *Proc. 2nd Symp. on Nonlinear Estimation Theory and Its Applications*, San Diego, California, pp. 51–58, 13–15 September (1971).
20. J. R. Rowland and W. M. Holmes, Statistical analysis techniques for error propagation in large scale missile systems, Tech. Rep. No. RG-TR-71-19, U. S. Army Missile Command, Redstone Arsenal, Alabama, August (1971).
21. R. J. Brown, Jr. and J. R. Rowland, Trajectory optimization for closed-loop nonlinear stochastic systems, *IEEE Trans. Automatic Control*, Vol. AC-17, No. 1, pp. 116–118 (1972).
22. R. J. Brown, Jr. and J. R. Rowland, Trajectory optimization for the nonlinear combined estimation and control problem, *Preprints of the IFAC Fifth World Congress*, Paper 32.6, Paris, France, 12–17 June (1972).

A SURVEY OF DIRECT NOISE PROPAGATION TECHNIQUES FOR LARGE-SCALE NONLINEAR SYSTEMS

Vijayendra M. Gupta
Graduate Research Associate
School of Electrical Engineering
Oklahoma State University
Stillwater, Oklahoma 74074

James R. Rowland
Associate Professor
School of Electrical Engineering
and Center for Systems Science
Oklahoma State University
Stillwater, Oklahoma 74074

Abstract

Direct methods for handling noise propagation problems in large-scale nonlinear systems are examined from the viewpoint of computability and efficiency. Comparisons are made between a fixed configuration method, the covariance analysis describing function technique, and the variational covariance algorithm. Initially, the different techniques are described with a particular emphasis on their advantages and disadvantages for large-scale nonlinear systems. Thereafter, a combination of the techniques is applied to a thirty-third order air defense missile system. The Monte Carlo simulation technique is then used to establish the validity of the numerical results for the combined direct algorithm.

Introduction

Early work on noise propagation in dynamical systems focused on the use of the Monte Carlo technique in which large numbers of simulation runs were ensemble-averaged to obtain statistical results. Since these Monte Carlo runs were often performed on the digital computer because of accuracy considerations, the basic problems were (1) the digital generation of a sequence of pseudo-random numbers to serve as a random input to the given system, (2) the sampling problem inherent in representing continuous systems and signals digitally, and (3) the determination of the number of simulation runs needed for acceptable statistical accuracy. Chambers [1] developed mixed congruential and multiplicative recurrence formulas for generating pseudo-random numbers on the digital computer. The optimal discrete representation of continuous input signals has been considered in [2]. It is shown in [3] and [4] that at least 1,000 simulation runs are required for statistical accuracies on the order of two per cent in certain applications. A more modern approach to the noise propagation problem is based on computing the desired statistical information directly. The new approach has resulted in several direct algorithms which are particularly amenable to digital computation based on accuracy, computational speed, computer storage, and algorithm complexity.

This paper presents a state-of-the-art survey of direct noise propaga-

Proceedings of the Seventeenth Midwest Symposium on Circuit Theory, Lawrence, Kansas, May 23-24, 1974.

tion techniques for large-scale nonlinear systems. Comparisons are made between a fixed configuration method, the covariance analysis describing function algorithm, and the variational covariance algorithm. After examining the relative merits of the three direct methods, a combined algorithm is applied to provide useful results for a thirty-third order system.

System Description

Consider a nonlinear dynamical system described by

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u}(t), \underline{w}(t), \underline{\beta}, t) \tag{1}$$

where $\underline{x}$ is the n-dimensional state vector, $\underline{u}(t)$ is an r-vector of non-random inputs, $\underline{w}(t)$ is an m-vector of random processes, $\underline{\beta}$ is an ℓ-vector of random bias inputs, and t is the independent variable representing time.

The input noise vectors $\underline{w}(t)$ and $\underline{\beta}$ have mean values specified by $\underline{\eta}_w$ and $\underline{\eta}_\beta$ and covariances matrices $\underline{Q}_w(t)$ and $\underline{Q}_\beta$, respectively. These may be defined mathematically as

$$\begin{aligned} E\{\underline{w}(t)\} &\triangleq \underline{\eta}_w(t) \\ E\{\underline{\beta}\} &\triangleq \underline{\eta}_\beta(t) \\ E\{(\underline{w}(t) - \underline{\eta}_w(t))(\underline{w}(\tau) - \underline{\eta}_w(\tau))^T\} &\triangleq \underline{Q}_w(t)\,\delta(t-\tau) \\ E\{(\underline{\beta} - \underline{\eta}_\beta)(\underline{\beta} - \underline{\eta}_\beta)^T\} &\triangleq \underline{Q}_\beta \end{aligned} \tag{2}$$

where $\delta(\cdot)$ represents the impulse function.

The problem is to utilize direct noise propagation techniques to obtain statistical information about the system state.

The Fixed Configuration Method

The fixed configuration method developed by Zirkle and Clark [5] is an extension of deterministic variational methods to stochastic systems. Described as a variational-averaging technique, this method requires that an initial assumed solution be an explicit function of time with parameters being random variables. The selection should be made such that statistical properties of the assumed solution are approximately the same as the statistical properties of the system response.

Zirkle and Clark assumed a solution of the form

$$\underline{\hat{x}}(t) = \underline{\hat{x}}(R,t) \tag{3}$$

where R is a j by k matrix of random variables used in approximating the system response. Their criterion for selecting R was

$$\int_{t_1}^{t_2} [f_j(\underline{x},\underline{u}(t),\underline{w}(t),\underline{\beta},t) - \dot{\hat{x}}_j]\, \frac{\delta \hat{x}_j}{\delta R_{jk}}\, \delta R_{jk} dt = 0 \tag{4}$$

for $j = 1, \ldots, n$ and $k = 1, \ldots, m$, where t_1 and t_2 indicate the specific time interval of interest. As an example, Zirkle and Clark considered $\ddot{x} + \omega_0^2 x + \epsilon x^3 = F(t)$ with an assumed form in (3) of $\hat{x}(t) = R \cos \omega t$. The input F(t) was a zero-mean, unity-variance, Gaussian, periodically stationary random process. Equation (4) became

$$\int_{iT}^{(i+1)T} [(\omega_0^2 - \omega^2)R \cos \omega t + \epsilon R^3 \cos^3 \omega t - F(t)]\delta R \cos \omega t \, dt = 0 \tag{5}$$

where $T = 2\pi/\omega$. The resulting algebraic equation was

$$\frac{3}{4} \epsilon R^3 + (\omega_0^2 - \omega^2) R = C \tag{6}$$

where C is the average of F(t) cos ωt over the period of interest T. Therefore, the probability density function of C and the nonlinear transformation in (6) could be used to determine the probability density function of R and, hence, the desired result in (3). Zirkle and Clark reported an error of less than 9% in the mean-squared value of the response amplitude for $\omega_0 = 1$, $\omega = 0.6$, and $\epsilon = 1/16$.

The main disadvantage is the problem of choosing the form of the assumed solution, which may be overcome for a particular application by a preliminary knowledge of the physical system behavior [6]. Moreover,it is quite difficult to implement this algorithm for large-scale systems on the digital computer. The primary advantage is that the complete state probability density function is available from the procedure.

The Describing Function Method

Another direct method for noise propagation is the covariance analysis describing function technique, which utilizes a statistical linearization of a given nonlinearity subject to pre-specified (usually Gaussian) input waveforms [7,8]. The result yields a quasilinear approximation of the transfer function of the nonlinearity, which is then used in the well-known covariance propagation equation for linear systems.

The differential equations for the mean $\underline{x}_N(t)$ and covariance matrix of P(t) of the quasilinear system state are

$$\begin{aligned} \dot{\underline{x}}_N &= N_{\underline{x}_N}(\underline{x}_N, P)\, \underline{x}_N + n_{\underline{w}} \\ \dot{P} &= AP + PA^T + Q_{\underline{w}} \end{aligned} \tag{7}$$

where $N_{\underline{x}_N}(\underline{x}_N, P)$ and A are matrix describing functions for the mean and random signals. These matrices are defined as

$$\begin{aligned} N_{\underline{x}_N}(\underline{x}_N, P)\, \underline{x}_N &= E\{\underline{f}(\underline{x}, t)\} \\ A &= E\{\underline{f}(\underline{x}, t)\, \underline{\delta x}^T\}\, P^{-1} \end{aligned} \tag{8}$$

where it has been assumed that the state $\underline{x}$ is the sum of its deterministic mean $\underline{x}_N$ and a random part $\underline{\delta x}$. The formulation in (7) treats the system

$$\dot{\underline{x}} = \underline{f}(\underline{x},t) + \underline{w}(t) \tag{9}$$

rather than the more general system in (1).

The advantage is that nonlinear effects are utilized in a linearized framework for a fast and efficient calculation of the covariance matrix associated with the system variables. The main disadvantage is that large-signal linearization techniques are applied to average statistical information about the nonlinearity. The describing function utilizes the nonlinear elements directly to yield noise propagation results, whereas the fixed configuration method requires an assumed form of the system response over a given time period. A third approach based on linearized incremental variations about nominal operating conditions is examined in the following section.

Variational Covariance Algorithm

The third method to be considered is the variational covariance algorithm which uses sufficiently small variations about the noise-free solution. The coefficients of the linearization matrices are updated during each integration interval whien applied to large-scale nonlinear systems. This technique was applied by Kuhnel and Sage [9] for sensitivity equations about a nominal flight path due to trajectory initial condition dispersions and random system variations. The direct and adjoint methods were used by Irwin and Hung [10] for evaluating the state covariance algorithm for large-scale, nonlinear dynamical systems.

It is assumed that the input noise disturbances cause sufficiently small deviations $\underline{\delta x}(t)$ about the (noise-free) nominal solution $\underline{x}_N(t)$ to permit linearization. Expanding (1) in a Taylor series about $\underline{x}_N(t)$ and neglecting higher-order terms above the first yields

$$\dot{\underline{\delta x}}(t) = A(t)\underline{\delta x}(t) + B(t)\underline{\delta\omega}(t) + C(t)\underline{\delta\beta} \tag{10}$$

where $\underline{\delta\omega}$ and $\underline{\delta\beta}$ are deviations from their respective means, and A(t), B(t), and C(t) are defined as the first partial derivatives of $\underline{f}(\cdot)$ with respect to $\underline{x}$, $\underline{\omega}$, and $\underline{\beta}$, respectively. These derivatives are evaluated at the nominal conditions in each case. The resulting variational covariance algorithm is given by

$$\begin{aligned}\dot{P}(t) = {} & A(t)P(t) + P(t)A^T(t) + B(t)Q_{\underline{w}}(t)B^T(t) \\ & + C(t)\, Q_{\underline{\beta}}H^T(t) + H(t)Q_{\underline{\beta}}C^T(t)\end{aligned} \tag{11}$$

where P(t) is the state covariance matrix and H(t) is the integral of the weighting pattern associated with C(t).

Rowland and Holmes [4] showed that the variational covariance algorithm can be applied to mildly nonlinear systems with acceptable results by using linearized incremental equations about the noise-free solution. This basic algorithm tends to yield unsatisfactory results for highly nonlinear systems, but the technique may be combined with the other methods described in this paper for acceptable results.

A Combined Direct Algorithm

The variational covariance algorithm has been combined with the describing function approach to yield improved noise propagation results for large-scale systems. Such a technique is useful for handling state-dependent switching nonlinearities. The input density function to the nonlinearity is assumed to be Gaussian, and the output density is determined by known nonlinear transformation methods. The variance of the output signal may then be calculated directly from the resulting non-Gaussian density function.

A thirty-third order six degree-of-freedom air defense missile system has been investigated [11]. The system includes a fifteenth-order autopilot, twelfth-order airframe equations with missile rotational and translational variables and launcher dynamics, fourth-order actuators, and a second-order seeker. Only certain segments of the missile flight could be handled by the combined algorithm because of severe nonlinearities. During this part of the flight, two relay nonlinearities in the seeker prohibited the variational covariance algorithm from giving acceptable results. However, the combined direct algorithm yielded results which compared favorably with twenty-five Monte Carlo ensemble-averaged runs. The seeker relay nonlinearity outputs were discrete levels, and the output variance was easily computed for the given operating conditions along the flight path. Finally, it should be noted that the combined direct algorithm gave unacceptable results for certain parts of the flight because the severe nonlinearities occurring in several of the missile sybsystems were not processed by using the describing function concepts.

Conclusions

Three direct noise propagation techniques have been examined, and a combined direct algorithm has been developed for large-scale applications. The fixed configuration method was shown to be difficult to implement for large-scale systems because of the requirement of an assumed form of the solution. The describing function method employed a statistical linearization of system nonlinearities with Gaussian input waveforms. Its application to large-scale systems requires a catalog of describing functions for the particular nonlinearities present in a given system. The variational covariance algorithm utilizes linearized variations about nominal operating conditions to yield acceptable results for mildly nonlinear systems. Moreover, the variational algorithm is easily extendable for stochastic filtering applications where the system state is to be estimated from a noise-corrupted measurement.

The combined direct algorithm was applied to a thirty-third order air defense missile system. Certain harsh nonlinearities were handled by the describing function approach and the other milder nonlinearities by the small-signal, incremental linearization approach. These numerical results compared favorably with the Monte Carlo simulation results obtained for the same large-scale system.

References

1. R. P. Chambers, "Random Number Generation", IEEE Spectrum, Vol. 4, No. 2, pp. 48-56, February 1967.

2. James R. Rowland, "Optimal Digital Simulations for Random Linear Systems with Integration Constraints", Computers and Electrical Engineering, Vol. 1, No. 1, pp.111-118, June 1973.

3. R. S. Bucy, "Realization of Nonlinear Filters", Proc. of the Second Symposium on Nonlinear Estimation Theory and Its Applications, San Diego, California, pp. 51-58, September 13-15, 1971.

4. James R. Rowland and Willard M. Holmes, "A Direct Covariance Algorithm for Computer-Aided Statistical Electronic Circuit Design", Int. Journal of Electronics, Vol. 19, No. 5, May, 1974 (To appear).

5. L. D. Zirkle and L. G. Clark, "A Variational Method for Approximating the Response of Nonlinear Stochastic Systems", Technical Report No. 1083, The University of Texas, Austin, Texas, August 1969.

6. V. H. Sumaria, "Response Statistics of Nonlinear Dynamical Systems Subjected to Narrowband Random Excitation", Master's Thesis, Department of Electrical Engineering, Oklahoma State University, May, 1973.

7. A. Gelb and W. E. Vander Veld, Multiple-Input Describing Functions and Nonlinear System Design, McGraw-Hill Book Company, 1968.

8. A. Gelb and R. S. Warren, "Direct Statistical Analysis of Nonlinear Systems -- CADET," Paper No. 72-875, AIAA Guidance and Control Conference, Stanford, California, August 1972.

9. W. C. Kuhnel and A. P. Sage, "Terminal State Error Analysis Using Adjoint-Generated Sensitivities", IEEE Transactions on Aerospace and Electronic Systems, Vol. 5, No. 2, pp. 185-194, March 1969.

10. J. D. Irwin and J. C. Hung, "Methods for Injection-Error Analysis and Their Comparison", IEEE Transactions on Automatic Control, Vol. 12, No. 3, pp. 276-281, June 1967.

11. V. M. Gupta, "An Efficient Covariance Matrix Implementation for Large-Scale Systems", Master's Thesis, Department of Electrical Engineering, Oklahoma State University, May 1973.

A STOCHASTIC ALGORITHM FOR SENSITIVITY ANALYSIS

James R. Rowland, Member, IEEE
School of Electrical Engineering and
Center for Systems Science
Oklahoma State University
Stillwater, Oklahoma 74074

Harold L. Pastrick, Member, IEEE
Guidance and Control Directorate (AMSMI-RGN)
U. S. Army Missile Research,
Development and Engineering Laboratory
U. S. Army Missile Command
Redstone Arsenal, Alabama 35809

Abstract

A direct stochastic sensitivity analysis algorithm is developed for linear dynamical systems having incompletely known input statistics. The new algorithm extends previous results by applying covariance propagation concepts which utilize as a forcing function the sensitivity covariance matrix associated with the uncertainty in the elements of the system input covariance matrix itself. The developed algorithm is evaluated in the context of a generalized sensitivity analysis formulation involving nonlinear transformations on the input signals. Numerical results are provided to demonstrate the usefulness of the new algorithm.

INTRODUCTION

Noise disturbances are inherent in all large-scale dynamical systems, typically appearing as a portion of the input signal, measurements, and/or variations in system parameters. Analysis of noise disturbance effects on the system has been accomplished primarily by representing the noise as a random process in systems modeled as being continuous or as a random sequence in discretely modeled systems [1,2]. Interest in the propagation of a random process through a large-scale dynamical system has centered on quantizing its effect on performance and ultimately on determining methods by which the effect can be reduced. The traditional approach on noise propagation problems has focused on the use of Monte Carlo

---

Manuscript Received: March 20, 1974. Submitted as a regular technical paper in the IEEE Transactions on Aerospace and Electronic Systems.

The work described here was supported by a Department of the Army project on Computer-Aided Design in Engineering (CAD-E), No. IE 762708 A090, and by Contract DAAH01-72-C-0672 between Oklahoma State University and the U. S. Army Missile Command.

techniques in which a large number of computer simulation runs are ensemble-averaged to obtain statistical results [3,4]. A more modern approach computes the effects of noise by solving the differential equation defining the state covariance matrix in terms of system parameters and the covariance of the input noise. Though well known and widely discussed as a technique for linear, time-varying systems [5-7], the method has also been applied to mildly nonlinear systems by use of appropriate linearization schemes. In particular, Irwin and Hung [8], Kuhnel and Sage [9], and Rowland and Holmes [10,11] have presented results for aerospace and electronic systems applications.

The covariance analysis method can be characterized by its requirement for a description of input noise statistics. However, in many cases those statistics are not well defined or, at best, they may be known only to within some tolerance level of uncertainty. The question arises regarding the usefulness of the covariance analysis method when a complete probabilistic description of the input process is not available. To this end sensitivity analysis, developed primarily for studies of filtering techniques [12,13], is needed to provide a useful method for determining the effects of errors in modeling input signal covariance matrices.

In this paper a new algorithm for sensitivity analysis is developed for linear dynamical systems where input statistics are not well known. The direct covariance propagation concept for linear systems with specified stochastic inputs is extended by considering variations in input noise statistics. Error analysis techniques based on specified input covariance matrices are reviewed initially for background information. A direct stochastic sensitivity analysis algorithm is then developed by expressing these covariance matrix equations in vector form and applying error propagation concepts to the resulting vector equation. A generalized sensitivity analysis formulation is presented to establish the validity of the new sensitivity algorithm. Brief examples are considered throughout the paper, but more complete numerical results are reserved for a separate section following the algorithm development.

PRELIMINARY ERROR ANALYSIS CONSIDERATIONS

As a basis for the main results to be developed later, consider the linear, time-varying, dynamical system represented by the vector differential equation

$$\underline{\dot{x}}(t) = A(t)\, \underline{x}\,(t) + B(t)\, \underline{w}(t) \tag{1}$$

where $\underline{x}$ is an n-dimensional plant state vector, $\underline{w}$ is an m-dimensional disturbance vector, and A and B are n by n and n by m system matrices, respectively. Let $\underline{w}(t)$ be a vector of white noise processes with mean $\mu_{\underline{w}}(t)$, and let the covariance matrix associated with $\underline{w}(t)$ be defined by

$$E\{[\underline{w}(t) - \mu_{\underline{w}}(t)]\,[\underline{w}(\tau) - \mu_{\underline{w}}(\tau)]^T\} = Q_{\underline{w}}(t)\,\delta(t-\tau) \tag{2}$$

where $\delta(\cdot)$ is the Dirac delta function.

Let P(t) represent the state covariance matrix, i.e.

$$P(t) \triangleq E\{[\underline{x}(t) - \mu_{\underline{x}}(t)]\,[\underline{x}(t) - \mu_{\underline{x}}(t)]^T\} \tag{3}$$

where $\mu_{\underline{x}}(t)$, the mean of $\underline{x}(t)$, may be determined from (1) by replacing $\underline{w}(t)$ by $\mu_{\underline{w}}(t)$ and $\underline{x}(t)$ by $\mu_{\underline{x}}(t)$. It has been shown that P(t) satisfies the matrix differential equation [1,2,5,7] given by

$$\dot{P}(t) = A(t)\ P(t) + P(t)\ A^T(t) + B(t)\ Q_{\underline{w}}(t)\ B^T(t) \tag{4}$$

This result is sometimes referred to as the direct covariance algorithm [11].*

Suppose $Q_{\underline{w}}(t)$ in (2) and (4) is not known exactly but lies somewhere on the bounded range between $Q_{\underline{w}_1}(t)$ and $Q_{\underline{w}_2}(t)$. The corresponding values of P(t) from (4) may be calculated to yield $P_1(t)$ and $P_2(t)$. Such an error analysis based on deterministic variations from some nominal conditions, such as $Q_{\underline{w}}(t) = Q_{\underline{w}_N}(t)$ and $P(t) = P_N(t)$, may be simplified when bounded variations $\delta Q_{\underline{w}}(t)$ occur above and below $Q_{\underline{w}_N}(t)$, i.e.

$$(Q_{\underline{w}_N} - \delta Q_{\underline{w}}) \leq Q_{\underline{w}} \leq (Q_{\underline{w}_N} + \delta Q_{\underline{w}}). \tag{5}$$

The resulting differential equation for $\delta P(t)$ is given by

$$\delta\dot{P}(t) = A(t)\delta P(t) + \delta P(t)A^T(t) + B(t)\delta Q_{\underline{w}}(t)B^T(t) \tag{6}$$

Therefore, P(t) varies between $P_1(t) = P_N(t) - \delta P(t)$ and $P_2(t) = P_N(t) + \delta P(t)$ for the variations of $Q_{\underline{w}}$ specified in (5).

Example 1

Suppose a steam-driven piston is used to impart a starting velocity condition to aircraft on a carrier deck. The steam pressure after each firing varies randomly on the piston. By neglecting the aircraft dynamics,

---

*It should be observed that the covariance results of this paper are applicable to linear systems and, hence, are not dependent upon the mean value of $\underline{w}(t)$. However, extensions are possible for an approximate analysis of mildly nonlinear systems, for which the coefficient matrices A(t) and B(t) are, in general, affected by $\mu_{\underline{w}}(t)$. These extensions are discussed in a later section of the paper.

it may be shown that the piston motion can be modeled by a first-order linear system of the form

$$\dot{x} = -a\ x + b\ w(t) \tag{7}$$

where the state x is the piston velocity, a is the ratio of the drag coefficient through the slotted rail guide to the piston mass, and b is the product of the pressure difference and the piston area-to-mass ratio. Random variations in steam pressure are assumed to be a Gaussian white noise signal w(t) with a constant variance Q_w. If Q_w is originally set at Q_{w_N} and then varied in both directions by a fixed amount δQ_w, the problem is to determine the resulting variations in the state covariance P(t).

The direct covariance algorithm in (4) may be used to propagate the nominal value of $Q_w(t)$ to yield

$$P_N(t) = P_N(0)e^{-2at} + \frac{b^2 Q_{w_N}}{2a}\ (1 - e^{-2at}) \tag{8}$$

Variations about this nominal solution may be computed by using the deterministic error analysis procedure, which yields from (6)

$$\delta P(t) = \delta P(0)e^{-2at} + \frac{b^2(\delta Q_w)}{2a}\ (1 - e^{-2at}) \tag{9}$$

Comparisons are indicated in Figure 1 between these deterministic results in (8) and (9) and corresponding results from the stochastic sensitivity analysis algorithm to be developed in the next section. Numerical data for these curves were obtained for $a = b = Q_w = 1$, $\delta Q_w = 0.5$, and $P_N(0) = 0$.

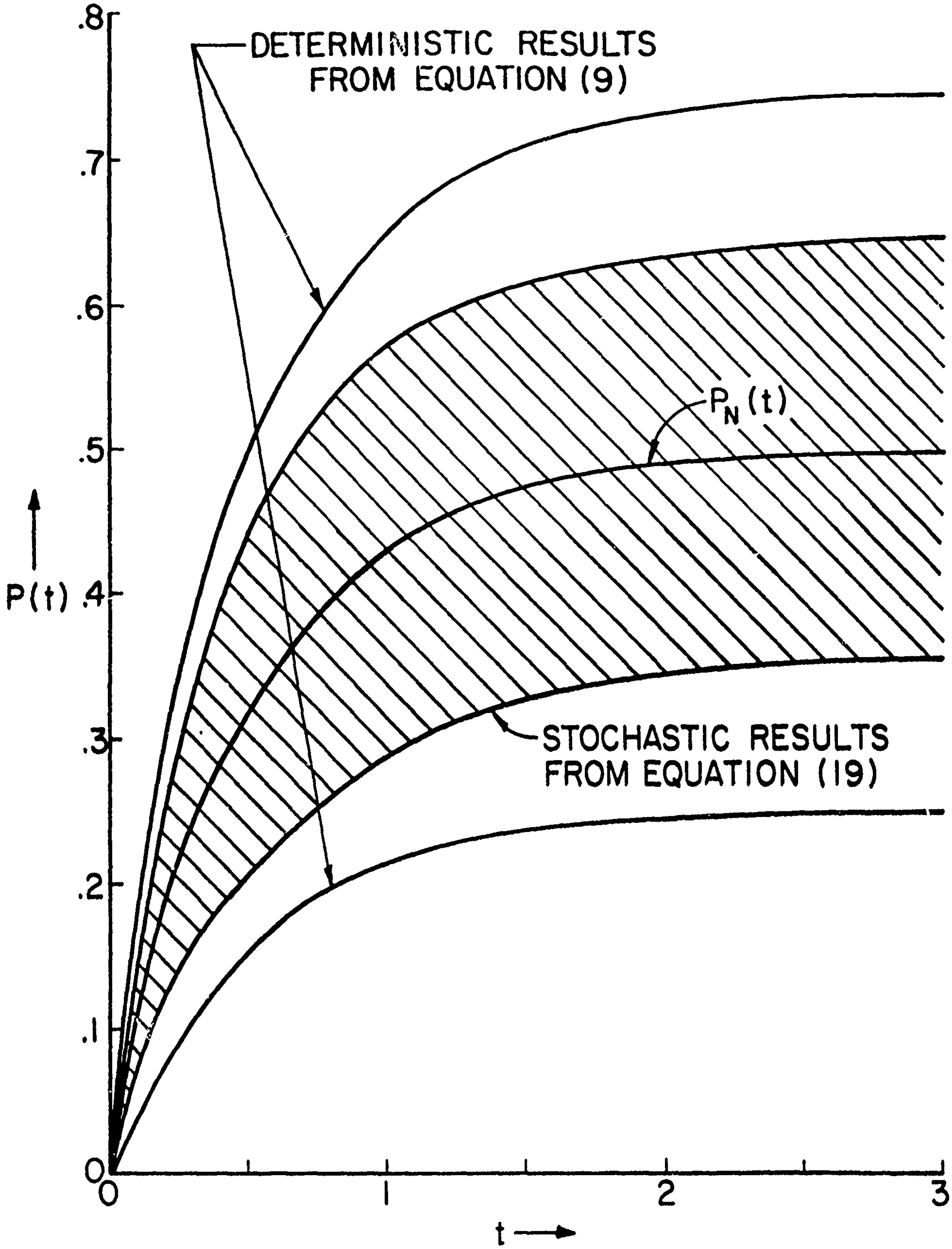


Figure 1. Comparison Between Stochastic and Deterministic Sensitivity Analysis Results for Examples 1 and 2.

STOCHASTIC SENSITIVITY ANALYSIS

The worst-case deterministic error analysis of the previous section may be expanded to provide results that are less conservative. Using the same techniques required for deriving (4), a similar error propagation equation for sensitivity analysis may be developed for deviations in P(t) due to stochastic variations in the input noise covariance matrix $Q_{\underline{w}}(t)$.

Let the matrix P(t) be expressed in terms of its column vectors $\underline{p}_j$ for j = 1, 2, ... , n as

$$P(t) = (\underline{p}_1, \underline{p}_2, \ldots, \underline{p}_j, \ldots, \underline{p}_n) \tag{10}$$

Therefore, one may form the vector $\underline{p}$ with n(n+1)/2 components as the distinguishable elements of P, i.e.

$$\underline{p} = \begin{pmatrix} [\underline{p}_1]_{U_1} \\ [\underline{p}_2]_{U_2} \\ \vdots \\ [\underline{p}_j]_{U_j} \\ \vdots \\ [\underline{p}_n]_{U_n} \end{pmatrix} \tag{11}$$

where the notation $[\underline{p}_j]_{U_j}$ denotes that only the upper j components of the vector $\underline{p}_j$ are retained in forming $\underline{p}$. Similarly, since $Q_{\underline{w}}$ is an m by m symmetric matrix, the vector $\underline{q}$ of dimension m(m+1)/2 may be formed as

$$\underline{q} = \begin{pmatrix} [\underline{q}_1]_{U_1} \\ [\underline{q}_2]_{U_2} \\ \vdots \\ [\underline{q}_m]_{U_m} \end{pmatrix} . \tag{12}$$

Let the covariance matrix of $\underline{q}$ be defined by

$$E\{[\underline{q}(t) - \mu_{\underline{q}}(t)]\,[\underline{q}(\tau) - \mu_{\underline{q}}(\tau)]^T\} = Q_{\underline{q}}(t)\delta(t-\tau) \tag{13}$$

where it is assumed that $\underline{q}(t)$ is a vector of white noise processes. Corresponding to the uncertainty in $Q_{\underline{w}}$, the covariance matrix associated with deviations in P may be expressed as

$$P_{\underline{p}}(t) = E\{[\underline{p}(t) - \mu_{\underline{p}}(t)]\,[\underline{p}(t) - \mu_{\underline{p}}(t)]^T\} \tag{14}$$

where $\mu_{\underline{p}}(t)$ is the vector of dimension $n(n+1)/2$ corresponding to a rearrangement of the elements of $P(t)$ from (4) with $Q_{\underline{w}}(t) = Q_{\underline{w}_N}(t)$.

Expressing $P(t)$ in terms of its column vectors as in (10) and expanding according to (4) yields for the jth column vector $\underline{p}_j$ the vector differential equation

$$\dot{\underline{p}}_j = A\underline{p}_j + \sum_{k=1}^{n} \underline{d}_k \underline{a}_j^T \underline{p}_k + B \sum_{k=1}^{m} \underline{d}_k \underline{b}_j^T \underline{q}_k \tag{15}$$

where $\underline{a}_j$ and $\underline{b}_j$ are n-vectors representing the jth columns of A and B, respectively, and $\underline{d}_k$ is defined as an n-vector with zero elements everywhere except for a single unity element in the kth row. Equation (15) may be expressed for all j between 1 and n in the vector-matrix form as

$$\dot{\underline{p}} = \Lambda \underline{p} + \Gamma \underline{q} \tag{16}$$

where repeated component differential equations in (15) have been omitted in a manner similar to that used in forming $\underline{p}$ in (11). The matrices Λ and Γ are $n(n+1)/2$ by $n(n+1)/2$ and $n(n+1)/2$ by $m(m+1)/2$, respectively. Applying error propagation concepts as in (4), the matrix differential equation for the sensitivity covariance matrix $P_{\underline{p}}(t)$ is

$$\dot{P}_{\underline{p}}(t) = \Lambda(t)P_{\underline{p}}(t) + P_{\underline{p}}(t)\Lambda^T(t) + \Gamma(t)Q_{\underline{q}}(t)\Gamma^T(t) \tag{17}$$

which is the main result of this paper.

<u>Example 2</u>

Let the scalar system (7) of Example 1 have a Gaussian white noise input w(t) with a covariance matrix Q_w which is uniformly distributed on the range $(Q_{w_N} - \delta Q_w, Q_{w_N} + \delta Q_w)$. The problem is to apply the stochastic sensitivity analysis algorithm (17) to determine corresponding variations in P(t).

The stochastic sensitivity analysis equation in (17) for this scalar example becomes

$$\dot{P}_p(t) = -4aP_p(t) + b^4 Q_q \tag{18}$$

where Q_q for the given uniformly distributed random process may be easily computed as $(\delta Q_w)^2/3$. Therefore, the solution of (18) is

$$P_p(t) = P_p(0)\, e^{-4at} + \frac{b^4(\delta Q_w)^2}{12a} \quad (1 - e^{-4at}) \tag{19}$$

Figure 1 compares these sensitivity results with those in (9) for the parameter values specified in Example 1. In particular, the one-sigma band $P_N(t) \pm \sqrt{P_p(t)}$ is shown for the stochastic algorithm. While this comparison is interesting, it should be recognized that two different situations are being considered in Examples 1 and 2. In Example 1, the error δQ_w, i.e. the variation of Q_w from Q_{w_N}, is known exactly. The resulting deterministic analysis yields the exact variation in P(t) from $P_N(t)$. On the other hand, the stochastic problem in Example 2 has a randomly (uniformly) distributed Q_w over a given range. Consequently, the one-sigma band on P(t) about its nominal may be determined according to the stochastic sensitivity analysis algorithm in (17).

A GENERALIZED SENSITIVITY APPROACH

It is instructive to reconsider the problem of the last section in the more general context of nonlinear transformations at the system input. If the uncertainty in $Q_{\underline{w}}(t)$ is due to the presence of a second white noise process r-vector, the nominal covariance matrix $Q_{\underline{w}_N}(t)$ must be determined from the joint probability density function of $\underline{w}(t)$ and $\underline{s}(t)$. It should be observed that the resulting $Q_{\underline{w}_N}(t)$ may be different than that obtained previously under the assumption that $\underline{s}(t)$ is non-random. If $Q_{\underline{w}_N}(t)$ is different, then (4) may be applied to yield a new nominal state covariance matrix $P_N(t)$, which includes elements due to the propagation effects of the modified $\underline{w}(t)$. Moreover, the sensitivity analysis procedure described earlier remains valid if variations about the new $P_N(t)$ are considered.

Suppose the joint probability density function relating the components of $\underline{w}$ and $\underline{s}$ is given by

$$f_{\underline{ws}}(\underline{\omega}, \underline{s}) = f_{\underline{w}|\underline{s}}(\underline{\omega}|\underline{s} = \underline{s})\, f_{\underline{s}}(\underline{s}) \tag{20}$$

Let $Q_{\underline{w}_N}(t)$ be defined at any time t as

$$Q_{\underline{w}_N} \triangleq E\{[\underline{w} - \mu_{\underline{w}}]\, [\underline{w} - \mu_{\underline{w}}]^T\}$$

$$= \int_{\mathcal{s}} \int_{\omega} (\underline{\omega} - \mu_{\underline{w}})\, (\underline{\omega} - \mu_{\underline{w}})^T f_{\underline{w}|\underline{s}}(\underline{\omega}|\underline{s} = \underline{s})\, d\underline{\omega}\, d\underline{s} \tag{21}$$

where the inner integral denotes an m-fold integration over the m components of $\underline{\omega}$ and the outer integral an r-fold integration over the r components of $\underline{s}$. Moreover, let $Q_{\underline{w}}$ be a matrix of random variables at any time t defined as

$$Q_{\underline{w}} = \int_{\omega} (\underline{\omega} - \mu_{\underline{w}})\, (\underline{\omega} - \mu_{w})^T f_{\underline{w}|\underline{s}}(\underline{\omega}|\underline{s} = \underline{s})\, d\underline{\omega} \tag{22}$$

It follows from (21) and (22) that $Q_{\underline{w}_N} = E\{Q_{\underline{w}}\}$, which may be evaluated as

$$Q_{\underline{w}_N} = E\{Q_{\underline{w}}\} = \int_{\mathcal{s}} Q_{\underline{w}}\, f_{\underline{s}}(\underline{s})\, d\underline{s} \tag{23}$$

because the uncertainty in $Q_{\underline{w}}$ is assumed to be due to the randomness of $\underline{s}(t)$. The resulting $Q_{\underline{w}_N}$ is different from that which would have been obtained from (22) by replacing $\underline{s}$ by its mean $\mu_{\underline{s}}$. Therefore, the covariance matrix associated with the uncertainty of $Q_{\underline{w}}$, i.e. $Q_{\underline{q}}(t)$, must be computed by using $f_{\underline{s}}(\underline{\delta})$ as shown in the following example.

Example 3

Consider the system (7) of Example 1 with a scalar white noise input w(t) which is uniformly distributed on the range $(\mu_w - s, \mu_w + s)$. Let s(t) be a second uniformly distributed white noise process on the range $(\mu_s - \alpha, \mu_s + \alpha)$, where μ_s and α are positive constants. The problem is to determine the nominal state covariance matrix $P_N(t)$ and the sensitivity analysis variations about that nominal as a function of time.

Since s(t) and $Q_w(t)$ are not identical in this example, the nominal variance of w(t) will be different than the value which would have been obtained by assuming that s(t) is non-random, i.e. $s(t) \equiv \mu_s$. For later reference, this value is given by

$$Q_{w_N} = E\{(w-\mu_w)^2\} = \int_{\mu_w-\mu_s}^{\mu_w+\mu_s} (\omega-\mu_w)^2 \frac{1}{2\mu_s} d\omega = \frac{\mu_s^2}{3} \tag{24}$$

and the resulting expression for $P_N(t)$ from (4) would have been

$$P_N(t) = P_N(0)\, e^{-2at} + \frac{b^2\mu_s^2}{6a} (1-e^{-2at}) \tag{25}$$

The correct $Q_{w_N}(t)$ may be determined from (21) as

$$Q_{w_N} = E\{(w-\mu_w)^2\} = \int_{\mu_s-\alpha}^{\mu_s+\alpha}\int_{\mu_w-\delta}^{\mu_w+\delta} (w-\mu_w)^2\left(\frac{1}{2\delta}\right)\left(\frac{1}{2\alpha}\right) dw\,d\delta \tag{26}$$

which yields

$$Q_{w_N} = \frac{\mu_s^2}{3} + \frac{\alpha^2}{9} \tag{27}$$

From (22) ,

$$Q_w = \int_{\mu_w-\delta}^{\mu_w+\delta} (w-\mu_w)^2\, f_{w|s}\,(w|s=\delta)\; dw = \frac{\delta^2}{3} \tag{28}$$

Therefore, the variance of Q_w, denoted by Q_q, may be calculated as

$$Q_q = E\{(Q_w-Q_{w_N})^2\} = \int_{\mu_s-\alpha}^{\mu_s+\alpha} [\delta^2/3 - \left(\frac{\mu_s^2}{3} + \frac{\alpha^2}{9}\right)]^2 \frac{1}{2\alpha}\, d\delta$$

$$= \frac{4}{27}\,\alpha^2\,(\mu_s^2 + \frac{\alpha^2}{15}) \tag{29}$$

Using (27) and (29), the corresponding values of $P_N(t)$ from (4) and $P_p(t)$ from (17) are

$$P_N(t) = P_N(0)e^{-2at} + \frac{b^2}{2a}\left(\frac{\mu_s^2}{3} + \frac{\alpha^2}{9}\right)(1-e^{-2at}) \tag{30}$$

and

$$P_p(t) = P_p(0)e^{-4at} + \frac{b^4}{4a}\left[\frac{4}{27}\alpha^2\,(\mu_s^2 + \frac{\alpha^2}{15})\right](1-e^{-4at}) \tag{31}$$

The results in (30) and (31) are plotted in Figure 2 for $a = b = \mu_s = 1$, $\alpha = 0.5$, and $P_N(0) = P_p(0) = 0$. Also included for comparison purposes is a plot of $P_N(t)$ for the case where $s(t)$ is assumed to be non- random

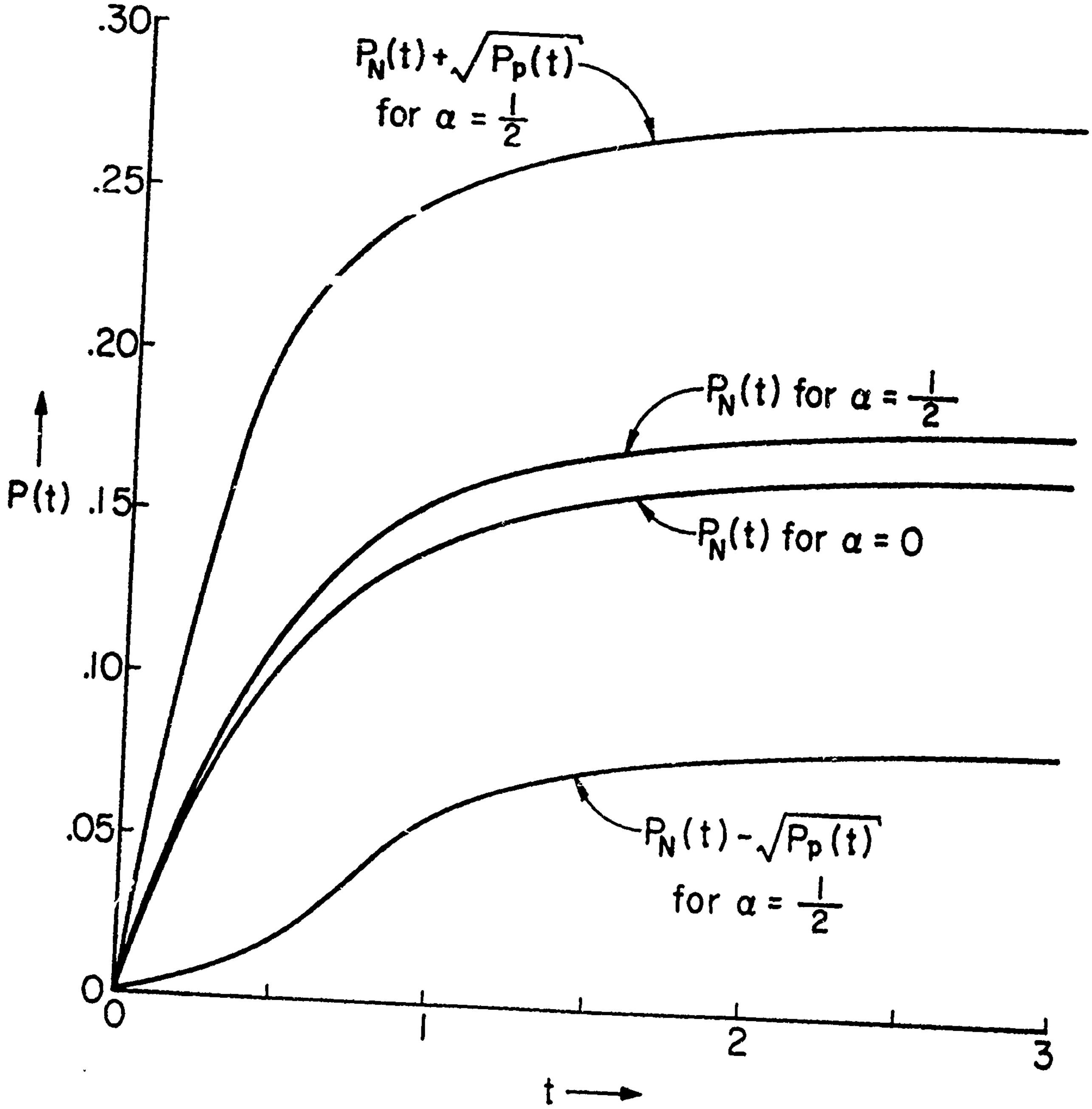


Figure 2. Stochastic Sensitivity Analysis Results for Example 3.

(α=0). This example demonstrates the importance of determining the correct Q_{w_N} and Q_q for use in error propagation and sensitivity analysis studies for general aerospace and electronic systems applications.

NUMERICAL RESULTS

Consider the second-order linear electronic circuit shown in Figure 3 and described mathematically by

$$\dot{v}_1 = -\frac{1}{R_1C_1} v_1 + \frac{1}{R_1C_1} w(t)$$

$$\dot{v}_2 = -\frac{1}{R_2C_2} v_2 + \frac{K}{R_2C_2} v_1 \qquad (32)$$

where $R_1C_1 = 1$, $R_2C_2 = 1/2$, and $K = 1/2$. The source voltage w(t), applied for all $t \geq 0$, is a zero-mean Gaussian white noise process with an incompletely specified variance Q_w. The uncertainty in Q_w is directly attributable to the fact that the standard deviation of w(t), denoted by s(t), is also a Gaussian white noise process. The mean of s(t) is 1.0 and its variance 0.1. The problem is to determine the propagation effects on the voltages across the capacitors, i.e. $v_1(t)$ and $v_2(t)$,

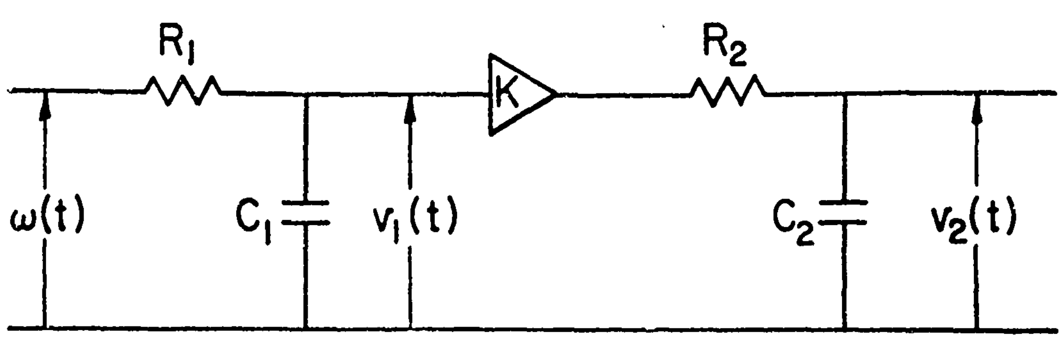


Figure 3. A Schematic Diagram of the Second-Order Linear Electronic Circuit Described by Equation (32)

due to the given white noise input w(t).

The value of Q_{w_N} for use in (4) may be determined from (23) as

$$Q_{w_N} = E\{Q_w\} = \int_{-\infty}^{+\infty} Q_w f_s(s)\, ds$$

$$= \int_{-\infty}^{+\infty} s^2 f_s(s)\, ds = \sigma_s^2 + \mu_s^2$$

where the variance Q_w has been replaced by the square of the standard deviation s(t). The expression in (33) yields the second moment of s, which is equivalent to the sum of its variance and the square of its mean. The component equations in (16) corresponding to (4) with $Q_w = Q_{w_N}$ may be written as

$$\dot{p}_{N_{11}} = 2a_{11}p_{N_{11}} + 2a_{12}p_{N_{12}} + Q_{w_N}$$

$$\dot{p}_{N_{12}} = a_{21}p_{N_{11}} + (a_{11}+a_{22})\, p_{N_{12}} + a_{12}p_{N_{22}} \tag{34}$$

$$\dot{p}_{N_{22}} = 2a_{21}p_{N_{12}} + 2a_{22}p_{N_{22}}$$

The given resistor and capacitor values for the system in (32) yields $a_{11} = -1$, $a_{12} = 0$, $a_{21} = 1$, and $a_{22} = -2$.

The value of Q_q for the stochastic sensitivity analysis may be determined as

$$Q_q = E\{(Q_w - Q_{w_N})^2\} = E\{Q_w^2\} - Q_{w_N}^2$$

$$= E\{s^4\} - Q_{w_N}^2 \tag{35}$$

$$= 2\sigma_s^2\,(\sigma_s^2 + 2\mu_s^2)$$

where the evaluation in (35) has been performed by expanding $E\{(s-\mu_s)^2\} = \sigma_s^2$, $E\{(s-\mu_s)^3\} = 0$, and $E\{(s-\mu_s)^4\} = 3\sigma_s^4$ and then substituting for $E\{s^4\}$ as indicated. Therefore, the component equations in (17) for $\dot{P}_{\underline{p}}(t)$ become

$$
\begin{aligned}
\dot{p}_{p_{11}} &= 4a_{11}\, p_{p_{11}} + 4a_{12}\, p_{p_{12}} + Q_q \\
\dot{p}_{p_{12}} &= a_{21}\, p_{p_{11}} + (3a_{11} + a_{22})\, p_{p_{12}} + a_{12} p_{p_{13}} + 2a_{12} p_{p_{22}} \\
\dot{p}_{p_{22}} &= 2a_{21}\, p_{p_{12}} + 2(a_{11}+a_{22})\, p_{p_{22}} + 2a_{12}\, p_{p_{23}} \\
\dot{p}_{p_{13}} &= 2a_{21}\, p_{p_{12}} + 2\,(a_{11} + a_{22})\, p_{p_{13}} + 2a_{12}\, p_{p_{23}} \\
\dot{p}_{p_{23}} &= a_{21}\, p_{p_{13}} + 2a_{21}\, p_{p_{22}} + (a_{11}+3a_{22})\, p_{p_{23}} + a_{12}\, p_{p_{33}} \\
\dot{p}_{p_{33}} &= 4a_{21}\, p_{p_{23}} + 4a_{22}\, p_{p_{33}}
\end{aligned}
\tag{36}
$$

Numerical results are shown in Figure 4 for the equations in (34) and (36). In particular, it is demonstrated that the one-sigma bands from (36) about the nominal noise propagation results from (34) vary considerably in magnitude. The bands for $p_{11}(t)$, $p_{12}(t)$, and $p_{22}(t)$ were determined as $p_{N_{11}}(t) \pm \sqrt{p_{p_{11}}(t)}$, $p_{N_{12}}(t) \pm \sqrt{p_{p_{22}}(t)}$, and $p_{N_{22}}(t) \pm \sqrt{p_{p_{33}}(t)}$, respectively. The other components of $P_{\underline{p}}(t)$ were used to determine the correlation between the band thicknesses in Figure

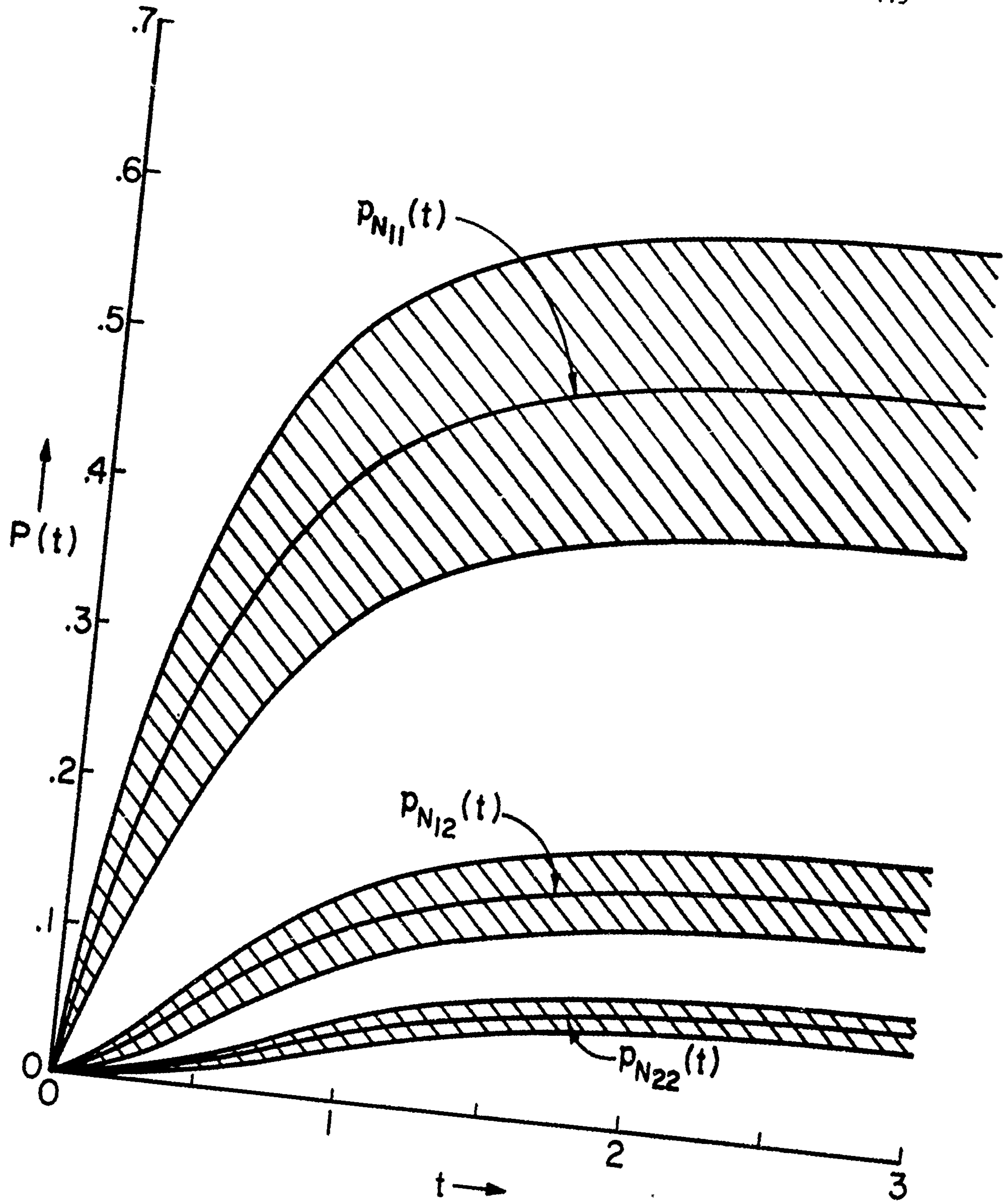


Figure 4. Stochastic Sensitivity Analysis Results for the Circuit of Figure 3.

4. Correlation coefficients were defined as

$$
\begin{aligned}
\rho_{12} &\stackrel{\Delta}{=} p_{p_{12}}(t) \Big/ \sqrt{p_{p_{11}}(t)\, p_{p_{22}}(t)} \\
\rho_{13} &\stackrel{\Delta}{=} p_{p_{13}}(t) \Big/ \sqrt{p_{p_{11}}(t)\, p_{p_{33}}(t)} \qquad (37) \\
\rho_{23} &\stackrel{\Delta}{=} p_{p_{23}}(t) \Big/ \sqrt{p_{p_{22}}(t)\, p_{p_{33}}(t)}
\end{aligned}
$$

Starting at slightly higher values for $t = 0$, these coefficients decreased monotonically to approximately 0.77, 0.56, and 0.93, respectively, after $t = 1$. Therefore, there exists a strong correlation between the thicknesses of the one-sigma bands for the given circuit in Figure 3.

DISCUSSION AND EXTENSIONS

A Gaussian assumption on the input signal $\underline{w}(t)$ is not required for the validity of the stochastic sensitivity analysis algorithm, although such signals frequently occur in practice. When the components of $\underline{w}(t)$ are jointly Gaussian, the resulting probability density function of the linear system state $\underline{x}$ (t) is also jointly Gaussian and, hence, may be written explicitly in terms of P(t) and the state mean $\mu_{\underline{x}}(t)$. Moreover, if $\underline{w}(t)$ is an m-vector of Gaussian colored noise signals, then an appropriately designed shaping filter may be utilized to yield an equivalent higher-order linear system having a Gaussian white noise input. In those cases where either $\underline{s}$ or $\underline{w}$ is a random bias signal, i.e. random variable, the noise propagation algorithm must be modified accordingly [11].

The stochastic sensitivity analysis algorithm may be applied for an approximate analysis of mildly nonlinear systems by considering linearized incremental variations about nominal operating conditions [10,11]. In such cases the nominal trajectory $\underline{x}_N(t)$ is obtained by replacing $\underline{w}(t)$ and $\underline{x}(t)$ in the nonlinear system equations by $\mu_{\underline{w}}(t)$ and $\underline{x}_N(t)$, respectively. The resulting time function $\underline{x}_N(t)$ is treated as an approximate estimate of the mean value of the system state $\underline{x}(t)$. A Taylor series expansion of $\underline{x}(t)$ about $\underline{x}_N(t)$ in terms of the variations $\underline{\delta x}(t)$ is truncated after first-order terms. Neglecting second and higher-order terms is reasonable if the dynamical system is mildly nonlinear. Such linearization schemes in filtering applications, where the nonlinear system state is observed in the presence of additive measurement noise, has led to the variational and extended Kalman filtering algorithms in common use today [14]. An extension of the stochastic sensitivity analysis principle to these filtering applications should yield some immediate useful results.

Finally, it is worthwhile to consider the similarities and differences between the concepts developed here and those utilized in [15]. An improved digital integration algorithm for mildly nonlinear systems was derived in [15] by considering variations upon variations about the current state. The similar concept of stochastic variations in $Q_{\underline{W}}$ upon stochastic variations in the input signal $\underline{w}(t)$ has been used in developing the algorithm of this paper. A major difference in the two applications is that exact integration results were obtained for linear systems in [15] by using a single variation, and further variations yielded no new information. On the other hand, a stochastic variation upon a stochastic variation provided useful exact sensitivity results in the present paper. Primarily of theoretical value, an extension analogous to the higher-order deterministic variations for nonlinear systems in [15] would be the consideration of higher-order stochastic variations ad infinitum in the input noise statistics and the tolerances on their specification.

CONCLUSIONS

A direct stochastic algorithm has been developed in this paper to provide sensitivity analysis information for linear systems with input statistics which are random. The elements of the input signal covariance matrix have been treated as white noise processes with known statistics and covariance propagation concepts applied to yield the new algorithm for determining stochastic variations in the state covariance matrix about its nominal. Numerical results for a second-order system have been presented to demonstrate the computations required in using the algorithm.

REFERENCES

[1] A. E. Bryson, Jr. and Y. C. Ho, Applied Optimal Control, Waltham, Mass.: Blaisdell Publishing Company, 1969.

[2] A. P. Sage and J. L. Melsa, Estimation Theory, New York: McGraw-Hill Book Company, 1971.

[3] F. J. Mullen, "Digital Simulation of Space Missions for Monte Carlo Analysis," Simulation, Volume 11, pp. 133-144, September 1968.

[4] A. R. Hurtubise, "Sample Sizes and Confidence Intervals Associated With a Monte Carlo Simulation Model Possessing a Multinomial Output," Simulation, Volume 13, pp. 71-77, February 1969.

[5] J. S. Meditch, Stochastic Optimal Linear Estimation and Control, New York: McGraw-Hill Book Company, 1969.

[6] K. J. Astrom, Introduction to Stochastic Control Theory, New York: Academic Press, 1970.

[7] A. P. Sage, Optimum Systems Control, Englewood Cliffs, New Jersey: Prentice-Hall, 1968.

[8] J. D. Irwin and J. C. Hung, "Methods for Injection-Error Analysis and Their Comparison," IEEE Transactions on Automatic Control, Volume AC-12, No. 3, pp. 276-281, June 1967.

[9] W. C. Kuhnel and A. P. Sage, "Terminal State Error Analysis Using Adjoint-Generated Sensitivities," IEEE Transactions on Aerospace and Electronic Systems, Volume 5, No. 2, pp. 185-194, March 1969.

[10] James R. Rowland and Willard M. Holmes, "Statistical Analysis Techniques for Error Propagation in Large-Scale Missile Systems," Technical Report No. RG-TR-71-19, U. S. Army Missile Command, Redstone Arsenal, Alabama, August 1971.

[11] James R. Rowland and Willard M. Holmes, "A Direct Covariance Algorithm for Computer-Aided Statistical Electronic Circuit Design," International Journal of Electronics, Vol. 36, No. 5, May 1974 (To Appear).

[12] T. Nishimura, "On the A-priori Information in Sequential Estimation Problems," IEEE Transactions on Automatic Control, Volume AC-11, No. 2, pp. 197-204, April 1966.

[13] R. E. Griffin and A. P. Sage, "Large and Small Scale Sensitivity Analysis of Optimum Estimation Algorithms," IEEE Transactions Automatic Control, Volume AC-13, No. 4, pp. 320-329, August 1968.

[14] M. Athans, "The Role and Use of the Stochastic Linear-Quadratic Gaussian Problem in Control System Design," IEEE Transactions on Automatic Control, Volume AC-16, No. 6, pp. 529-552, December 1971.

[15] James R. Rowland and Willard M. Holmes, "A Variational Approach to Digital Integration," IEEE Transactions on Computers, Vol. C-20, No. 8, pp. 894-900, August 1971.

APPENDIX B

COMPUTER PROGRAM FOR THE STANDARD METHOD OF MONTE CARLO SIMULATION

The program for the Monte Carlo technique using the standard method has been included in this Appendix. A second-order system was used to obtain Monte Carlo results for 25, 50, 100, 200, 500 and 1000 runs for comparing the results with other methods as discussed in Chapter II.

Statements 35 through 46 were used to generate zero-mean, unity-variance, Gaussianly distributed random numbers. Subsequent instructions were used for the calculation of the output variance and the percentage error on the output variance. The Runge-Kutta second-order formula (RK2) was used for integrating the second-order system.

```
      DIMENSION XE(2),XS(2),XM0(2),XM1(2),S(10),SOL(10),DIF(10),XEM(10)
      T=0.
      H = 0.05
      MS=2
      II = 0
      NTOTAL=100
      MTOT=NTOTAL/10
      DO 31 N=1,MTOT
      S(N) = 0.
      XEM(N)= 0.
 31   CONTINUE
      XMEAN=0.
      IX=31571
      DUM=0.1
      SIG = SQRT(1./H)
      DO 82 I=1,40
      IF(I.EQ.1) GO TO 81
      IF(I.EQ.2) GO TO 81
      IF(I.EQ.4) GO TO 81
      IF(I.EQ.8) GO TO 81
      IF(I.EQ.20)GO TO 81
      IF(I.EQ.40)GO TO 81
      GO TO 82
 81   NUM = 25*I
      XNUM = NUM
      XNUM1 = XNUM*XNUM
      XNUM2 = XNUM - 1.0
      XNUM3 = XNUM/XNUM2
      JJ=II+1
      DO 32 M=JJ,NUM
      XE(1)=0.
      XE(2)=0.
      DO 42 N=1,MTOT
      DO 52 L=1,10
      IY=19971*IX
      IYP=IY/1048576
      IX=IY-IYP*1048576
      AX=IX
      U=AX/1048576.
      IF(U)5,5,6
 5    U=-U
 6    CONTINUE
      IX=IY
      Z=SQRT(-2.0*ALOG(DUM))*SIG
      XNORM =Z*COS(6.28318*U)+XMEAN
      DUM=U
      CALL XEQN(XE,XM0,XNORM)
      DO 23 K=1,MS
 23   XS(K)=XE(K)+H*XM0(K)
      CALL XEQN(XS,XM1,XNORM)
      DO 24 K=1,MS
 24   XE(K)=XE(K)+0.5*H*(XM0(K)+XM1(K))
 52   CONTINUE
      S(N) = S(N) + XE(1)*XE(1)
```

```
      XEM(N) = XEM(N) + XE(1)
   42 CONTINUE
   32 CONTINUE
      WRITE(6,84)NUM
   84 FORMAT(1X,//' NO. OF RUNS = ',I5)
      WRITE(6,15)
      WRITE(6,83)
   83 FORMAT(T11,'TIME',T25,'S(NA)',T38,'SOL(NA)',T53,'DIF(NA)',T68,
     1'XEM(NA)')
      DO 62 NA=1,MTOT
      XNA=NA
      T=H*XNA*10.
      SOL(NA)   =0.08333333333-0.5*EXP(-2.0*T)+0.6666666667*EXP(-3.0*T)
     1-0.25*EXP(-4.0*T)
      XEM(NA) = XEM(NA)*XEM(NA)/(XNUM*XNUM)
      XEM(NA) = XEM(NA)*XNUM3
      S(NA) = S(NA)/XNUM2- XEM(NA)
      DIF(NA) = 100.0*(S(NA)-SOL(NA))/SOL(NA)
      WRITE(6,7)T,S(NA),SOL(NA),DIF(NA),XEM(NA)
    7 FORMAT(10X,F5.2,4F15.6)
   62 CONTINUE
      WRITE(6,15)
   15 FORMAT(//)
      SS1=0.
      DO 97 NA=1,MTOT
      SS1=SS1+ABS(DIF(NA))
      S(NA) = (S(NA)+XEM(NA))*XNUM2
      XEM(NA) = SQRT(XEM(NA)/XNUM3)*XNUM
   97 CONTINUE
      S1=SS1*0.1
      WRITE(6,94)S1
   94 FORMAT(20X,'PER CENT ERROR = ',F20.8)
      II=NUM
   82 CONTINUE
      STOP
      END

      SUBROUTINE XEQN(XD,XMD,RT)
       DIMENSION XD(2),XMD(2)
      XMD(1)=XD(2)
      XMD(2)=-2.0*XD(1)-3.0*XD(2)+RT
      RETURN
      END
```

APPENDIX C

THE COMPUTER SOFTWARE PACKAGE APPLIED TO THE LARGE-SCALE MISSILE SYSTEM

This appendix includes the implemented computer software package on the thirty-third order math model of a six degree-of-freedom air defense missile system. In addition to the modification of the original program, the nine subprograms which have been implemented are COEFF, COVAR, RUNGKP, MDERIV, SNOISE, DETARA, INTA2M, RANDU, and RANDG.

The main program initializes all the covariance matrix elements and other variables used in the program. Subroutine INTA2M initializes the coefficient matrix elements. The SYSINT subprogram updates the nonlinear terms of the coefficient matrix, enters Subprogram COEFF to evaluate the coefficients for the implicitly related variables, and calls the COVAR subprogram where the covariance differential equations are calculated. These equations are then integrated by entering RUNGKP from SYSINT. Subroutines SNOISE and DETARA are used to calculate the variance of the noise introduced in the SEEKER program.

A listing of all subroutines is provided on the following page to indicate their location within this appendix.

```
C ***
C      TERMINAL HOMING - ALL DIGITAL SIMULATION
C ***
C
C *** BLANK COMMON HOUSES AERODYNAMIC COEFFICIENTS AND DERIVATIVES IN
C *** TABULAR FORM FOR USE BY THE 1, 2, AND 3 VARIATE LOOK UP SCHEME.
C
       COMMON DXDYDZ(60),IADD(20),AERO(1360)
C
C *** COMMON BLOCK /TIMES CONTAINS CURRENT TIME, STEP LENGTH AND OTHER
C *** EVENT TIMES IN THE SIMULATION.
C
       COMMON /TIMES/T,DT,TBC,TSTOP,IPR,J,LAUNCH
       DOUBLE PRECISION T,DT
C
C *** PROGRAM SELECTION (MODULE TEST OR SYSTEM RUN) AND MODULE TEST
C *** DATA(WHEN MODE=2)
C
       COMMON /CNTRL/MODE, MDLS(4),IV,DATAM(16,4)
C
C *** COMMON BLOCK /AUTOP/ CONTAINS INTEGRATION VARIABLES, DERIVATIVES
C *** AND INTERMEDIATE VARIABLES REQUIRED BY THE AUTOPILOT MODULE
C
       COMMON /AUTOP/NA,VA(15),DVA(15),OV(7)
C
C *** COMMON BLOCK /SEEKR/ CONTAINS INTEGRATION VARIABLES, DERIVATIVES
C *** AND INTERMEDIATE VARIABLES REQUIRED BY THE AUTOPILOT MODULE
       COMMON / SEEKR/ NS,VS(2),DVS(2),OSV(8)
C
C *** COMMON BLOCK /VANES/ CONTAINS INTEGRATION VARIABLES AND DERIVATIVES
C *** REQUIRED IN THE VANE ANGLE CALCULATION MODULE
C
       COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
C
C *** COMMON BLOCK /ROTATE/ CONTAINS ROTATIONAL VARIABLES AND DERIVATIVES
C *** USED IN THE MISSILE MODULE
C
       COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
      1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
C
C *** COMMON BLOCK /STATEV/ CONTAINS TRANSLATIONAL VARIABLES AND
C *** DERIVATIVES
C
       COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
C
C *** COMMON BLOCK /ADDV/   CONTAINS ADDITIONAL VARIABLES DERIVED FROM
C *** THE STATE (INTEGRATION) VARIABLES
C
       COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VSS,RHO
C
C *** COMMON BLOCK /COEFS/ CONTAINS THE THRUST AND AERODYNAMIC
C *** COEFFICIENTS AND DERIVATIVES  OBTAINED BY TABLE INTERPOLATION
C
       COMMON /COEFS/THR,AERC(18)
```

```
C
C *** COMMON BLOCK CONTAINS AIRFRAME CONSTANTS GOVERNING AERODYNAMIC
C *** FORCES AND THRUST MISALIGNMENT
C
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
C
C *** COMMON BLOCK /MSINCG/ CONTAINS MASS, INERTIAS AND CG POSITION OF
C *** THE AIRFRAME PLUS THE CONSTANT VALUES FROM WHICH THEY ARE OBTAINED
C
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
C
C *** COMMON BLOCK /FCEMOM/ CONTAINS THE AERODYNAMIC FORCES, MOMENTS,
C *** AND THRUST MISALIGNMENT COMPONENTS
C
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
C
C *** COMMON BLOCK /INCEPT/ CONTAINS TARGET POSITION AND VELOCITY,
C *** TARGET-MISSILE INTERCEPT SPEED AND RANGE AND INPUTS TO THE SEEKER
C
      COMMON /INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
C
C *** COMMON BLOCK /TRANSF/ CONTAINS MATRICES FOR CONVERSION FROM
C *** VARIOUS COORDINATE SYSTEMS TO OTHERS
C
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
C
C *** COMMON BLOCK CONTAINS UTILITY VALUES SUCH AS GRAVITY ACC. AND
C *** RADIANS TO DEGREES CONSTANTS.
C
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      COMMON / SEEKK/ SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS
      COMMON /UTILTY/G,RTD
      COMMON /VMG/ H,MS
      COMMON /VMG1/P1(33,33),DP8(33,33)
      COMMON /VMG9/JUNK,VTIME1,VTIME2,VNOISD,NUMM,NOMNAL
      COMMON /BLOK1/DTH
      COMMON /BLOCK1/P(33,33),DP(33,33),DP9(33,33)
      COMMON /BLOCK2/ A2(33,33),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON / BLOCK7/KK3,THRP,TIMP
      COMMON /BLOCK8/KK1,KK5,VP
      COMMON /BLOCK9/KOK,IS1
      COMMON /BLIK2/ AVD(4),BVD(4)
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS,VBEPS,VBEPSZ,VBEPSY
      COMMON /MBLOK1/KOUNT1,XNORM(4),S1(33,40)
      COMMON /MBLOK2/SIG1,DUM,XMEAN,IX,N1,I1,I2,K1,N2
      COMMON /MBLOK3/ S2(33,40)
      COMMON /MVMG/S3(40),KINTER,KONTER
      COMMON /MVMG1/JX,YNORM(33),DAMU,SIGU,XMEANU,IS2
      COMMON /MVMG2/TEPSTG(33),KIT,IKPR,TMVE,TMRNG ,EZTMP,EYTMP
      COMMON /MVMG3/S4(33)
      DIMENSION LBL(10),TRANFR(33)
```

```
C
C ****************************************************************************
C   VTIME1 -- CONTROLS  SWITCHING TIME FROM MONTE CARLO TO COVARIANCE PROGRAM
C   VTIME2 -- CONTROLS  SWITCHING TIME FROM COVARIANCE TO MONTE CARLO PROGRAM
C   VNOISD -- CONTROLS THE NOISE INPUT IN DEGREES IN SUBROUTINES - TARGET
C                AND SNOISE.
C   NUM -- CONTROLS THE NUMBER OF MONTE CARLO RUNS.
C   JUNK -- USED FOR PRINTING OUT A MATRIX ELEMENTS ONLY ONCE IN SUBR. SYSINT.
C   DTH -- USED AS A STEP SIZE FOR COVAR IN SYSINT SUBR.
C   KQUNT -- CONTROLS THE FREQUENCY OF CALCULATIONS OF A MATRIX COEFFICIENTS
C   KAT -- USED AS COUNTER FOR COVAR INTEGRATION ROUTINE.
C   KICK -- CONTROLS THE FREQUENCY OF PRINTOUT FOR THE COVARIANCE MATRIX
C   KK1, KK3, KK5,K400 -- USED IN COEFF SUBR. TO CONTROL THE CALCULATION.
C   B2(I) -- B MATRIX ELEMENTS USED IN COVAR SUBR. FOR CALCULATING P(1,1),P(4,4)
C   AVD(I), BVD(I) -- USED FOR CALCULATING NL 'A' MATRIX COEFFICIENTS IN
C                     SYSINT AND VANEMD SUBRS.
C   P1(I,K), DP8(I,K), DP9(I,K) -- USED AS TEMPORARY STORAGE FOR COVARIANCE
C                                  INTEGRATION
C   KIT -- USED AFTER  SWITCHING FROM COVAR TO MONTE CARLO PROGRAM.
C   IKPR -- USED TO PRESERVE THE VALUE OF IPR.
C   KINTER -- ATTAINS A VALUE OF NUM+1 IN MAIN AND DOES NOT CHANGE THEREAFTER.
C   KONTER -- USED IN SYSRUN AND INITIALIZED IN MAIN TO CONTROL SWITCHING FROM
C                COVAR TO MONTE CARLO PROGRAM AFTER VTIME2.
C   N1, K1 -- USED IN SYSINT TO CONTROL THE ENSEMBLE-AVERAGING INTERVAL.
C   IS2 -- USED TO CALCULATE RANDOM NUMBERS EQUAL TO THE ORDER OF THE SYSTEM.
C ****************************************************************************
C
       READ(5,1)SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS,
     1WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,PYAK1,PYBK1,PYIK1,
     2WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS,PB,QB,RB,UE,VE,WE,
     3THETA,PHI,PSI,X,Y,Z,S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE,SI,WO,
     4WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,VGAIN,VLIM,VRLIM
C
       WRITE(4) SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS,
     1WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,PYAK1,PYBK1,PYIK1,
     2WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS,PB,QB,RB,UE,VE,WE,
     3THETA,PHI,PSI,X,Y,Z,S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE,SI,WO,
     4WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,VGAIN,VLIM,VRLIM
C
C   **************************************************************************
C   TO RUN THE PROGRAM AS NOMINAL, COVARIANCE, OR MONTE CARLO OR THEIR
C   COMBINATIONS, USE THE FOLLOWING INITIALIZATIONS.
C     NOMINAL FLIGHT
C         VTIME1 = 0.0
C         VTIME2 = (THE VALUE OF 'TSTOP')
C         KINTER = ('NUM+1')
C         KONTER = ('NUM+1')
C         NOMNAL = 0
C     COVARIANCE PROGRAM
C         VTIME1 = 0.0
C         VTIME2 = (THE VALUE OF 'TSTOP')
C         KINTER = ('NUM+1')
C         KONTER = ('NUM+1')
C         NOMNAL = 1
```

```
C     MONTE CARLO PROGRAM
C         VTIME1 = (THE VALUE OF 'TSTOP')
C         VTIME2 = (THE VALUE OF 'TSTOP')
C         KINTER = 1
C         KONTER = 1
C         NOMNAL = 1
C  ****************************************************************************
C
      VTIME1 = 0.0
      VTIME2 = 12.02
      NUM = 25
      KINTER = 26
      KONTER = 26
      NOMNAL = 0
      VNOISD = 2.0
C
C  ****************************************************************************
C     NOMINAL -- MONTE CARLO PROGRAM
C         VTIME1 = 0.0
C         VTIME2 = (SPECIFY THE SWITCHING TIME)
C         KINTER = ('NUM+1')
C         KONTER = ('NUM+1')
C         NOMNAL = 0
C     COVARIANCE -- MONTE CARLO PROGRAM
C         VTIME1 = 0.0
C         VTIME2 = (SPECIFY THE SWITCHING TIME)
C         KINTER = ('NUM+1')
C         KONTER = ('NUM+1')
C         NOMNAL = 1
C     MONTE CARLO -- COVARIANCE PROGRAM
C         VTIME1 = (SPECIFY THE SWITCHING TIME)
C         VTIME2 = (THE VALUE OF 'TSTOP')
C         KINTER = 1
C         KONTER = 1
C         NOMNAL = 1
C     MONTE CARLO -- COVARIANCE -- MONTE CARLO PROGRAM
C         VTIME1 = (SPECIFY THE SWITCHING TIME)
C         VTIME2 = (SPECIFY THE SWITCHING TIME)
C         KINTER = 1
C         KONTER = 1
C         NOMNAL = 1
C  ****************************************************************************
C
      NUMM=NUM + 1
      JUNK = 1
      IS1 = 0
      DTH = 0.0
      EZNOIS = 0.0
      EYNOIS = 0.0
      VBEPS = 0.0
      VBEPSZ = 0.0
      VBEPSY = 0.0
      KIKK=0
      KIK1 = 0
```

```
      KOUNT = 10
      KAT = 0
      KICK = 40
      KIK = 1
      KOK = 0
      K400 = 0
      KK1 = 1
      KK3 = 0
      KK5 = 0
      VP = 1.0
      B2(1) = 6750.0
      B2(2) = 6750.0
      TMVEL = -0.10
      TMRNGE = 10000.1
      DO 88 I=1,4
      AVD(I) = 0.0
 88   BVD(I) = 0.0
      DO 29 I=1,MS
      DO 29 K=1,MS
      A2(I,K) = 0.
      DP(I,K) = 0.
      P1(I,K) = 0.0
      DP8(I,K) = 0.0
      DP9(I,K) = 0.0
 29   P(I,K) =0.
C   SUBROUTINE INTA2M IS USED TO INITIALIZE THE A MATRIX COEFFICIENTS
      CALL INTA2M
      READ(5,62)(AREA(I),I=1,30)
      AREA(31) = 0.0
C
C *** READ THRUST AND AERODYNAMIC TABLES FROM CARDS
C
      WRITE (6   ,900)
      KNT1 = 1
      KNT2 = 3
      IL = 1
   30 READ (   5,910) I,J,K,(DXDYDZ(L),L=KNT1,KNT2),LBL
      IF (I.EQ.999)GO TO 40
      WRITE (6   ,920) LBL
      KNT1 = KNT2+1
      L = KNT2/3
      IADD(L) = IL
      KNT2 = KNT2+3
      IF (J.EQ.0)J=1
      IF (K.EQ.0)K=1
      IU = I*J*K+IL-1
      READ (   5,930) (AERO(L),L=IL,IU)
      IL = IU+1
      GO TO 30
   40 CONTINUE
      G = 32.17
      RTD = 57.2957795
C
C *** CALL INITIA TO INITIALIZE THE PROGRAM AND READ RUN DATA
```

```
C
      CALL INITIA
C
      MS1 = 33
      XNUM = NUM
      XNUM1 = XNUM*XNUM
      XNUM2 = XNUM - 1.0
      XNUM3 = XNUM/XNUM2
      NTOTAL = 1200
      MTOT = NTOTAL/40
      IX = 31571
      DUM = 0.1
      JX = 28651
      DAMU = .12
      SIGU = 1.0
      XMEANU = 0.0
      KIT = 0
      IKPR = 40
      DO 1004 IS=1,15
 1004 YNORM(IS)  = 0.0
      DO 1005 IS=1,MS
 1005 TEPSTG(IS) = 0.0
      DO 31 K1=1,MS1
      DO 31 N1=1,40
      S2(K1,N1) = 0.0
   31 S1(K1,N1) = 0.
      DO 81 I=1,40
   81 S3(I) = 0.0
      DO 308 I=1,MS
      S4(I) = 0.0
  308 TRANFR(I) = 0.0
      XMEAN = 0.
      DO 32 M1 = 1,NUMM
      DO 33 I=1,4
   33 XNORM(I) = 0.0
      WP = PB*RTD
      WQ = QB*RTD
      WR = RB*RTD
      BTHETA = THETA*RTD
      BPH = PHI*RTD
      BPS = PSI*RTD
      TMVEL = -0.10
      TMRNGE = 10000.1
      N1 = 39
      K1 = 40
      KOUNT1= 0
C     ************************
      IF(VTIME1.EQ.0.0)GO TO 32
C     ************************
      NS = 2
      VS(1) = 0.
      VS(2) = 0.
      NT = 6
      NR = 6
```

```
      NA = 15
      DO 4 IS=1,15
    4 VA(IS) = 0.
      NV = 4
      DO 5 IS=1,4
    5 VV(IS) = 0.
      UT(1) = 0.
      UT(2) = 0.
      UT(3) = 0.
      XT(1) = 10000.
      XT(2) = 0.
      XT(3) = 0.
      REWIND 4
C
      READ(4)  SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS,
     1WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,PYAK1,PYBK1,PYIK1,
     2WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS,PB,QB,RB,UE,VE,WE,
     3THETA,PHI,PSI,X,Y,Z,S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE,SI,WO,
     4WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,VGAIN,VLIM,VRLIM
C
      DT = 0.25D-02
      IPR = 40
      CALL INTHRC
      CALL INTRAN
      CALL INAUPT
      IF(KINTER.EQ.NUMM)GO TO 32
C *** CALL TOTAL SYSTEM RUN CONTROL ROUTINE
C
      CALL SYSRUN
      KINTER = KINTER + 1
   32 CONTINUE
      IF(VTIME1.EQ.0.0)GO TO 306
      DO 302 I=1,MS
      DO 302 IM=1,MS
      DP9(I,IM) = S2(I,N2)*S2(IM,N2)*XNUM3/XNUM1
  302 DP8(I,IM) = DP8(I,IM)/XNUM2 - DP9(I,IM)
      P(32,32) = DP8(32,32)
      P(33,33) = DP8(33,33)
      DO 305 I=1,31
      DO 305 IM=1,31
      P(I,IM) = DP8(I,IM)
      DP8(I,IM) = 0.0
  305 DP9(I,IM) = 0.0
      DO 304 IM=1,MS1
      DO 303 N1=1,MTOT
      S2(IM,N1) = S2(IM,N1)*S2(IM,N1)*XNUM3/XNUM1
      S1(IM,N1) = S1(IM,N1)/XNUM2 - S2(IM,N1)
  303 CONTINUE
  304 CONTINUE
      DO 311 IM=1,MS1
  311 WRITE(6,202)IM,(S1(IM,N1),N1=1,MTOT)
      WRITE(6,988)(S3(I),I=1,MTOT)
      IF(VTIME1.GE.TSTOP)GO TO 307
  306 DO 36 I=1,4
```

```
36    XNORM(I) = 0.0
      CALL SYSRUN
      IF(VTIME2.GE.TSTOP)GO TO 307
C     ************************
      G1 = 0.0
      G2 = 0.0
      DP9(1,1) = SQRT(P(1,1))
      DO 101 I=2,MS
101   DP9(1,I) = P(1,I)/DP9(1,1)
      DO 102 I=2,MS
      K=I-1
      DO 103 IJ=1,K
103   G1 = G1 + DP9(IJ,I)*DP9(IJ,I)
      DP9(I,I) = SQRT(P(I,I)-G1)
      DO 105  JM=1,MS
      IF(JM.LE.I) GO TO 105
      DO 104 M1=1,K
104   G2 = G2 + DP9(M1,I)*DP9(M1,JM)
      DP9(I,JM) = (P(I,JM) - G2)/DP9(I,I)
105   CONTINUE
102   CONTINUE
C     *************************
      NUMN = NUM - 1
      DO 34 M1 = 1,NUMN
      DO 35 I=1,4
35    XNORM(I) = 0.0
      N1 = 39
      K1 = 40
      KOUNT1= 0
C *********************
      T = VTIME2
      DT = 0.0025
C *********************
      CALL INSYST
      CALL INRK4
      DO 114 IM=1,MS
114   TRANFR(IM) = TEPSTG(IM)
      IS2 = MS
      CALL RANDU
      DO 115 I=1,MS
      DO 115 IM=1,I
115   TRANFR(I) = TRANFR(I) + DP9(IM,I)*YNORM(IM)
      DO 113 I=1,15
113   VA(I) = TRANFR(I)
      UE = TRANFR(16)
      VE = TRANFR(17)
      WE = TRANFR(18)
      X  =TRANFR(19)
      Y  =TRANFR(20)
      Z  =TRANFR(21)
      PB = TRANFR(22)
      QB = TRANFR(23)
      RB = TRANFR(24)
      THETA = TRANFR(25)
```

```
      PHI= TRANFR(26)
      PSI= TRANFR(27)
      DO 3 IS = 1,4
    3 VV(IS) = TRANFR(IS+27)
      VS(1) = TRANFR(32)
      VS(2) = TRANFR(33)
      IPR = IKPR
      TMVEL = TMVE
      TMRNGE = TMRNG
      WP = PB*RTD
      WQ = QB*RTD
      WR = RB*RTD
      BTHETA = THETA*RTD
      BPH = PHI*RTD
      BPS = PSI*RTD
      OSV(1) = EZTMP
      OSV(2) = EYTMP
C
C *** CALL TOTAL SYSTEM RUN CONTROL ROUTINE
C
      CALL SYSRUN
C
   34 CONTINUE
      DO 203 IM=1,MS
      DO 204 N1 = 1,MTOT
      S2(IM,N1) = S2(IM,N1)*S2(IM,N1)*XNUM3/XNUM1
      S1(IM,N1) = S1(IM,N1)/XNUM2 - S2(IM,N1)
  204 CONTINUE
  203 CONTINUE
      DO 211 IM=1,MS
  211 WRITE(6,202)IM,(S1(IM,N1),N1=1,MTOT)
      WRITE(6,988)(S3(I),I=1,MTOT)
  307 STOP
    1 FORMAT(8(8F10.4/))
   62 FORMAT(10F8.6)
  202 FORMAT(//1X,'VAR(',I2,',N1) =',7E15.5/5(13X,7E15.5/))
  900 FORMAT (1H1, 50X,'T-H AERODYNAMIC TABLES')
  910 FORMAT (3I3, 1X,3F10.0, 10A4)
  920 FORMAT (/45X,10A4)
  930 FORMAT (8F10.0)
  988 FORMAT(/1X,'STIME =',10F10.5/8X, 10F10.5)
      END
```

CARD

```
      SUBROUTINE INITIA
C ***
C     THIS ROUTINE READS VARIOUS RUN DATA FROM CARDS AND INITIALIZES
C     THE REMAINDER OF THE PROGRAM
C ***
      COMMON /CNTRL/MODE,MDLS(4),IV,DATAM(16,4)
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J ,LAUNCH
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI
      COMMON /INCEPT/UT(3),XT(3)
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      DOUBLE PRECISION T,DT
      CALL INTHRC
      CALL INTRAN
      CALL INAUPT
      READ ( 5,900) MODE,MDLS,IV,IT,ITCG,IRAIL,IWIND
      GO TO(20,30),MODE
20    READ( 5,930) (DATAM(J,1),J=1,16),(DATAM(J,2),J=1,4)
      READ ( 5,940)DT,TSTOP,IPR
      IF(IV.NE.0)READ( 5,910)UE,VE,WE,X ,Y,Z,PB,QB,RB,THETA,PHI,PSI
      IF (IT.NE.0)READ( 5,910)UT,XT
      IF(ITCG.NE.0)READ( 5,910)XTCG,YTCG,ZTCG
      IF(IRAIL.NE.0)READ( 5,910)RL1,RL2
      IF(IWIND.NE.0)READ( 5,910)WUE,WVE,WWE
      RETURN
30    DO 40 I=1,4
      IF (MDLS(I).EQ.0)GO TO 40
      READ( 5,920) DATAM(1,I)
      READ( 5,910) (DATAM(J,I),J=2,16)
40    CONTINUE
      RETURN
900   FORMAT(16I5)
910   FORMAT(8F10.0)
920   FORMAT(F20.0)
930   FORMAT(20A4)
940   FORMAT(2F10.0,I10)
      END
```

```
      SUBROUTINE INTA2M
      COMMON /UTILTY/G,RTD
      COMMON /BLOCK2/ A2(33,33)
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      TMP1 = WQ1*WQ1
      TMP2 = 2.*DQ1*WQ1
      TMP3 = PYAK1*PYBK1
      TMP4 = PYAK1+PYBK1
      TMP5 = WQG*WQG
      TMP6 = 2.*DQG*WQG
      TMP7 = PYIK1*WQ1*WQ1/TMP3
C
C *** CONSTANT 'A' MATRIX ELEMENTS
C
      A2(1,1) = -3.*TAUZ
      A2(1,2) = TAUZ*A2(1,1)
      A2(1,3) =-TAUZ*TAUZ*TAUZ
      A2(2,1) = 1.
      A2(3,2) = 1.
      A2(4,4) = -3.*TAUY
      A2(4,5) = TAUY*A2(4,4)
      A2(4,6) = -TAUY*TAUY*TAUY
      A2(5,4) = 1.
      A2(6,5) = 1.
      A2(7,7) = -2.*DP1*WP1
      A2(7,8) = -WP1*WP1
      A2(7,26) = -A2(7,8)*RTD
      A2(8,7) = 1.
      A2(10,2) = -TMP7
      A2(10,3) = -TMP7*TAUL
      A2(10,5) = TMP7
      A2(10,6) = -A2(10,3)
      A2(10,10) =-TMP2
      A2(10,11) = -TMP1
      A2(10,23)=RTD*TMP7
      A2(10,24) =-A2(10,23)
      A2(11,10) = 1.
      A2(12, 2) = A2(10, 2)
      A2(12, 3) = A2(10, 3)
      A2(12, 5) = A2(10, 5)
      A2(12, 6) = A2(10, 6)
      A2(12,10) = TMP4+A2(10,10)
      A2(12,11) = TMP3+A2(10,11)
      A2(12,23) = A2(10,23)
      A2(12,24) = A2(10,24)
      A2(13,2) = -TMP7
      A2(13,3) = -TMP7*TAUL
      A2(13,5) = -TMP7
      A2(13,6) = A2(13,3)
      A2(13,13) = -TMP2
      A2(13,14) = -TMP1
      A2(13,23) = A2(12,23)
      A2(13,24) = A2(13,23)
```

```
      A2(14,13) = 1.
      A2(15, 2) = A2(13, 2)
      A2(15, 3) = A2(13, 3)
      A2(15, 5) = A2(13, 5)
      A2(15, 6) = A2(13, 6)
      A2(15,13) = TMP4+A2(13,13)
      A2(15,14) = TMP3+A2(13,14)
      A2(15,23) = A2(13,23)
      A2(15,24) = A2(13,24)
      A2(19,16) =1.0
      A2(20,17) =1.0
      A2(21,18) =1.0
      A2(26,22) = 1.0
      RETURN
      END
```

```
      BLOCK DATA
      COMMON / SEEKP/ NS,VS(2),DVS(2),DSV(8)
      COMMON / SEEKK/ SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS
      COMMON /AUTOP/NA,VA(15),DVA(15),OV(7)
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /VANES/NV,VV(4),DVV(4), DEL(3)
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      COMMON /VMG/ H,MS
      DATA H,MS/0.0025,33/
      DATA SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS/3.,3.,0.,
     10.05,0.,0.,0.,0./
      DATA NS,VS/ 2,2*0.0/
      DATA WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,PYAK1,PYBK1,
     1PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS/373.,1.,15.,15.,2.,
     26750.,12.,60.,130.,.53,.33,40.,15.,2.8,115.,.64,15.,7.,1.,0.0,0.0/
      COMMON /MSINCC/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WF,WQ,WR,BTHETA,BPH,BPS
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /UTILTY/G,RTD
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      DATA G,RTD/32.17,57.2957795/
      DATA NT,NR/6,6/
      DATA PB,QB,RB,UE,VE,WE,THETA,PHI,PSI,X,Y,Z/0.,0.,0.,.1,0.,0.,5.,
     10.,0.,0.,0.,-40./
      DATA NA,VA/15,15*0./
      DATA NV,VGAIN,VLIM,VRLIM/4,15.,20.,200./
      DATA VV/4*0./
      DATA S,D,XTCG,YTCG,ZTCG/.267,.584,2.75,0.,0./
      DATA RL1,RL2,WUE,WVE,WWE/3.5,6.07,0.,0.,0./
      DATA SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP/195.8,121.,19.4,.241,15.
     111,2.54,-.375,-.15/
      DATA UT/3*0./
      DATA XT/10000.,0.,0./
      END
```

```
      SUBROUTINE SYSINT
C **
C     THIS ROUTINE INTEGRATES ALL EQUATIONS OVER 1 TIME STEP
C ***
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J ,LAUNCH
      COMMON /STATEV/NT,VT(6),DVT(6)
      COMMON /ROTATE/NR,VR(6),DVR(6),SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI
     1,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /SEEKR/ NS,VS(2),DVS(2),OSV(8)
      COMMON /AUTOP/NA,VA(15),DVA(15),DVAD(7)
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY
     1,RLCG,RDCG
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RA2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /VMG/ H,MS
      COMMON /VMG1/P1(33,33),DP8(33,33)
      COMMON /VMG9/JUNK,VTIME1,VTIME2,VNOISO,NUMM,NOMNAL
      COMMON /BLOCK1/P(33,33),DP(33,33),DP9(33,33)
      COMMON /BLOCK2/ A2(33,33),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON /BLOCK4/ VV5(4),DLTC(4)
      COMMON /BLOK1/DTH
      COMMON /BLIK1/BPHISM
      COMMON /BLIK2/ AVD(4),BVD(4)
      COMMON /INCEPT/UT(3),XT(3),TMVEL,TMRNGE
      COMMON /MBLOK1/KOUNT1,XNORM(4),S1(33,40)
      COMMON /MBLOK2/SIG1,DUM,XMEAN,IX,N1,I1,I2,K1,N2
      COMMON /MBLOK3/ S2(33,40)
      COMMON /MVMG/S3(40),KINTER
      COMMON /MVMG2/TEPSTG(33),KIT,IKPR,TMVE,TMRNG ,EZTMP,EYTMP
      COMMON /MVMG3/S4(33)
      DOUBLE PRECISION T,DT, HALFDT
      DIMENSION QT(12),QR(12),QA(30),QV(8)
C
C     ************************
      IF(T.LT.VTIME2)GO TO 4
      IF(KIT.NE.0)GO TO 7
      KINTER = 1
      KIT = 1
      DO 1 IS = 1,15
    1 TEPSTG(IS) = VA(IS)
      DO 2 IS = 1,6
      TEPSTG(IS+15) = VT(IS)
    2 TEPSTG(IS+21) = VR(IS)
      DO 3 IS=1,4
    3 TEPSTG(IS+27) = VV(IS)
      TEPSTG(32) = VS(1)
      TEPSTG(33) = VS(2)
      IKPR = IPR
      TMVE = TMVEL
      TMRNG = TMRNGE
      EZTMP = OSV(1)
      EYTMP = OSV(2)
```

```
      WRITE(6,8)(TEPSTG(I),I=1,33),TMVE,TMRNG ,EZTMP,EYTMP ,T
8     FORMAT(8E15.6/4(8E15.6/))
C     ************************
C
4     IF(KINTER.EQ.NUMM)GO TO 191
7     DTT = SNGL(DT)
      SIG1= SQRT(1./DTT)
      I1 = 1
      I2 = 2
      CALL RANDG
      I1 = 3
      I2 = 4
      CALL RANDG
191   DO 40 KUT = 1,4
      GO TO (30,10,20,10),KUT
10    T = T+HALFDT
      GO TO (15,20),J
15    CALL THRCON
20    CALL AUTOPT
      CALL VANEMD
      CALL TRANSM
      CALL ROTATM
30    CALL RK4(NA,VA,QA,KUT)
      CALL RK4(NV,VV,QV,KUT)
      CALL RK4(NT,VT,QT,KUT)
      CALL RK4(NR,VR,QR,KUT)
  40  CONTINUE
      CALL AUTOPT
      CALL VANEMD
      CALL TRANSM
      CALL ROTATM
C
C **************
      IF(KINTER.NE.NUMM)GO TO 1001
      IF(T.LE.VTIME1)GO TO 1001
      IF(NOMNAL.EQ.0)GO TO 1001
C **************
C
C
C *** NONLINEAR 'A' MATRIX ELEMENTS
C
      IF(ABS(BPHISM).GE.( RLIM-0.001)) GO TO 12
      A2(9,7) = RK1*(RA1+RB2+A2(7,7))/RA1/RB2
      A2(9,8) = RK1*(1.+A2(7,8))/RA1/RB2
      A2(9,26) = RK1*A2(7,26)/RA1/RB2
      GO TO 13
12    A2(9,7) =0.0
      A2(9,8) =0.0
      A2(9,26) = 0.0
13    IF(ABS(VA(12)).GE.(PYLIM-0.001)) GO TO 22
      A2(28,12) = VGAIN
      A2(30,12) = VGAIN
      GO TO 23
22    A2(28,12) =0.0
```

```
CARD
          A2(30,12) =0.0
   23     IF(ABS(VA(15)).GE.(PYLIM-0.001)) GO TO 32
          A2(29,15) = VGAIN
          A2(31,15) = VGAIN
          GO TO 33
   32     A2(29,15) =0.0
          A2(31,15) =0.0
   33     IF(ABS(VV(1)) .GE.( VLIM-0.001)) GO TO 42
          A2(28,28) = -VGAIN
          GO TO 43
   42     A2(28,28) =0.0
   43     IF(ABS(VV(2)) .GE.( VLIM-0.001)) GO TO 52
          A2( 29,29) = -VGAIN
          GO TO 53
   52     A2(29,29) = 0.0
   53     IF(ABS(VV(3)) .GE.( VLIM-0.001)) GO TO 62
          A2(30,30) = -VGAIN
          GO TO 63
   62     A2(30,30) =0.0
   63     IF(ABS(VV(4)) .GE.( VLIM-0.001)) GO TO 72
          A2(31,31) =-VGAIN
          GO TO 73
   72     A2(31,31) =0.0
   73     CONTINUE
          IF(ABS(AVD(1)).GE.(VRLIM-0.001)) GO TO 83
          A2(28,7) = A2(9,7)*VGAIN
          A2(28,8) = A2(9,8)*VGAIN
          A2(28,9) = 0.1*VGAIN
          A2(28,26) =A2(9,26)*VGAIN
          GO TO 84
   83     A2(28,7) = 0.0
          A2(28,8) = 0.0
          A2(28,9) = 0.0
          A2(28,12) =0.0
          A2(28,26) =0.0
          A2(28,28) =0.0
   84     IF(ABS(AVD(2)).GE.(VRLIM-0.001)) GO TO 93
          A2(29,7) = A2(28,7)
          A2(29,8) = A2(28,8)
          A2(29,9) = A2(28,9)
          A2(29,26) = A2(28,26)
          GO TO 94
   93     A2(29,7) = 0.0
          A2(29,8) = 0.0
          A2(29,9) = 0.0
          A2( 29,15) =0.0
          A2(29,26) =0.0
          A2(29,29) =0.0
   94     IF(ABS(AVD(3)).GE.(VRLIM-0.001)) GO TO 103
          A2(30, 7) =-A2(28,7)
          A2(30, 8) =-A2(28,8)
          A2(30, 9) =-A2(28,9)
          A2(30,26) =-A2(28,26)
          GO TO 104
```

```
103   A2(30,7) = 0.0
      A2(30, 8) =0.0
      A2(30, 9) =0.0
      A2(30,12) =0.0
      A2(30,26) =0.0
      A2(30,30) =0.0
104   IF(ABS(AVD(4)).GE.(VRLIN-0.001)) GO TO 113
      A2(31, 7) =A2(30,7)
      A2(31, 8) =A2(30,8)
      A2(31, 9) =A2(30,9)
      A2(31,26) = A2(30,26)
      GO TO 114
113   A2(31,7) = 0.0
      A2(31, 8) =0.0
      A2(31, 9) =0.0
      A2(31,15) = 0.0
      A2(31,26) =0.0
      A2(31,31) =0.0
114   IF(BVD(1).LE.0.0)GO TO 133
      A2(28,7) = 0.0
      A2(28,8) = 0.0
      A2(28,9) = 0.0
      A2(28,12) =0.0
      A2(28,26) =0.0
      A2(28,28) =0.0
133   IF(BVD(2).LE.0.0)GO TO 143
      A2(29,7) = 0.0
      A2(29,8) = 0.0
      A2(29,9) = 0.0
      A2(29,15) =0.0
      A2(29,26) =0.0
      A2(29,29) =0.0
143   IF(BVD(3).LE.0.0)GO TO 153
      A2(30,7) = 0.0
      A2(30, 8) =0.0
      A2(30, 9) =0.0
      A2(30,12) =0.0
      A2(30,26) =0.0
      A2(30,30) =0.0
153   IF(BVD(4).LE.0.0)GO TO 163
      A2(31,7) = 0.0
      A2(31, 8) =0.0
      A2(31, 9) =0.0
      A2(31,15) = 0.0
      A2(31,26) =0.0
      A2(31,31) =0.0
163   CONTINUE
      A2(25,23) =CSPHI
      A2(25,24) =-SNPHI
      A2(27,23) = SNPHI/CSTHA
      A2(27,24) = CSPHI/CSTHA
      A2(26,23) = SNTHA*A2(27,23)
      A2(26,24) = SNTHA*A2(27,24)
      A2(25,26) = -VR(2)*SNPHI-VR(3)*CSPHI
```

```
      A2(26,25) = -A2(25,26)/(CSTHA*CSTHA)
      A2(27,26) = (-VR(3)*SNPHI+VR(2)*CSPHI)/CSTHA
      A2(26,26) =A2(27,26)*SNTHA
      A2(27,25) =-A2(26,25)*SNTHA
      IF(KOUNT.NE.10)GO TO 111
      KOUNT = 0
      CALL COEFF
C ********************
      IF(JUNK.EQ.0)GO TO 111
      DO 333 I=1,MS
333   WRITE(6,344)I,(A2(I,K),K=1,I)
      JUNK = 0
C ********************
111   DTH = SNGL(DT)
      DTH = DTH/2.0
      DO 222 IJ=1,2
      KAT = 1
      DO 222 IJ=1,2
      CALL COVAR
      CALL RUNGKP
222   KAT = KAT + 1
      DO 29 II=1,MS
      IF(P(II,II).GE.1.0E-10)GO TO 29
      DO 28 IJ=1,MS
      P(II,IJ) = 0.0
      P(IJ,II) = 0.0
 28   CONTINUE
29    CONTINUE
      IF(KICK.NE.40)GO TO 299
C *********
      WRITE(6,125)(VT(I),I=4,6)
C *********
      WRITE(6,124)T
      DO 288 I=1,MS
288   WRITE(6,11)I,(P(I,K),K=1,I)
      KICK = 0
299   CONTINUE
      KICK = KICK + 1
      KOUNT = KOUNT+1
344   FORMAT(//1X,'A(',I2,', J) =',7E15.5/4(11X,7E15.5/))
11    FORMAT(//1X,'P(',I2,', J) =',7E15.5/4(11X,7E15.5/))
124   FORMAT(1X,'TIME = ',F8.4)
125   FORMAT(' X=',E15.5,'Y=',E15.5,' Z=',E15.5)
C *********************************
1001  IF(KINTER.EQ.NUMM)GO TO 6
C *********************************
      N1 = N1 + 1
      IF(N1.NE.K1)GO TO 6
      K1 = N1 + 40
      N2 = N1/40
      DO 201 IM = 1,15
      S2(IM,N2) = S2(IM,N2) + VA(IM)
201   S1(IM,N2) = S1(IM,N2) + VA(IM)*VA(IM)
      DO 202 IM=1,6
```

```
      S2(IM+15,N2) = S2(IM+15,N2) + VT(IM)
202   S1(IM+15,N2) = S1(IM+15,N2) + VT(IM)*VT(IM)
      DO 203 IM=1,6
       S2(IM+21,N2) = S2(IM+21,N2) + VR(IM)
203    S1(IM+21,N2) = S1(IM+21,N2) + VR(IM)*VR(IM)
      DO 207 IM=1,4
       S2(IM+27,N2) = S2(IM+27,N2) + VV(IM)
207    S1(IM+27,N2) = S1(IM+27,N2) + VV(IM)*VV(IM)
      DO 208 IM=1,2
      S2(IM+31,N2) = S2(IM+31,N2) + VS(IM)
208   S1(IM+31,N2) = S1(IM+31,N2) + VS(IM)*VS(IM)
      IF(T.LT.(VTIME1 - 0.0025).OR.T.GE.VTIME2)GO TO 307
      DO 303 I=1,15
303   S4(I) = VA(I)
      DO 304 I=1,6
      S4(I+15) = VT(I)
304   S4(I+21) = VR(I)
      DO 305 I=1,4
305   S4(I+27) = VV(I)
      DO 306 I=1,2
306   S4(I+31) = VS(I)
      DO 301 I=1,MS
      DO 301 IM=1,MS
301   DP8(I,IM) = DP8(I,IM) + S4(I)*S4(IM)
307   T1 = SNGL(T)
      S3(N2) = T1
6     RETURN
      ENTRY INSYST
      HALFDT = .5D+0*DT
      RETURN
      END
```

```
      SUBROUTINE RANDG
      COMMON /MBLOK1/KOUNT1,XNORM(4),S1(33,40)
      COMMON /MBLOK2/SIG1,DUM,XMEAN,IX,N1,I1,I2,K1
C ***
C     THIS ROUTINE CALCULATES THE NORMALLY DISTRIBUTED RANDOM
C     NUMBERS 'XNORM'
C ***
      IY=19971*IX
      IYP=IY/1048576
      IX=IY-IYP*1048576
      AX=IX
      U=AX/1048576.
      IF(U.LE.0.0)U=-U
      IX=IY
      Z=SQRT(-2.0*ALOG(DUM))*SIG1
      XNORM(I1) = Z*COS(6.28318*U)+XMEAN
      XNORM(I2) = Z*SIN(6.28318*U)+XMEAN
      DUM=U
      RETURN
      END
```

```
      SUBROUTINE RANDU
C
C ******************************************************************
C THIS PROGRAM GENERATES YFL WHICH IS UNIFORMLY DISTRIBUTED BETWEEN 0 AND 1
C ******************************************************************
C
      COMMON /MVMG1/JX,YNORM(33),DAMU,SIGU,XMEANU,IS2
      DO 1 I=1,IS2
      JY = JX*65539
      IF(JY.LT.0)JY=JY+2147483647+1
      JX = JY
      YFL = JY
      YFL = YFL*0.4656613E-9
      Z = SQRT(-2.0*ALOG(DAMU)*SIGU)
      YNORM(I) = Z*COS(6.28318*YFL)+XMEANU
    1 DAMU = YFL
      RETURN
      END
```

```
      FUNCTION XLIMIT(V,VLIM)
      IF(ABS(V)-VLIM)40,40,10
   10 IF (V)20,30,30
   20 XLIMIT = -VLIM
      RETURN
   30 XLIMIT = VLIM
      RETURN
   40 XLIMIT = V
      RETURN
      END
```

CARD

```
      SUBROUTINE RK4(N,V,Q,K)
C ***
C     THIS ROUTINE INCREMENTS VARIABLES, GIVEN THEIR DERIVATIVES ACCORDING
C     TO THE RUNGE-KUTTA 4 POINT SCHEME.
C ***
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J1,LAUNCH
      DOUBLE PRECISION T,DT
      DIMENSION V(N),Q(N)
      DO 50 I=1,N
      J=N+I
      GO TO(10,20,30,40),K
   10 Q(J) = V(J)
      Q(I) = V(I)
      V(I) = V(I)+DTOV2*V(J)
      GO TO 50
   20 V(I) = Q(I)+DTOV2*V(J)
      Q(J) = Q(J)+V(J)+V(J)
      GO TO 50
   30 V(I) = Q(I)+DT1*V(J)
      Q(J) = Q(J)+V(J)+V(J)
      GO TO 50
 40   V(I) = Q(I)+DT1*(Q(J)+V(J))*0.1666667
   50 CONTINUE
      RETURN
      ENTRY INRK4
      DTOV2 = SNGL(DT*.5D+0)
      DT1 = SNGL(DT)
      RETURN
      END
```

CARD

```
      SUBROUTINE RUNGKP
      COMMON /VMG/ H,MS
      COMMON /VMG1/P1(33,33),DP8(33,33)
      COMMON /BLOCK1/P(33,33),DP(33,33)
      COMMON /BLOCK2/ A2(33,33),KIK,KOUNT,KICK,KAT
      COMMON /BLOK1/DTH
      GO TO(10,30),KAT
 10   DO 20 I=1,MS
      DO 20 J=1,I
      P1(I,J) = P(I,J)
      DP8(I,J) = DP(I,J)
 20   P(I,J) = P(I,J) + DTH*DP(I,J)
      RETURN
  30  VDT = DTH/2.0
      DO 40 I=1,MS
      DO 40 J=1,I
 40   P(I,J) = P1(I,J) + VDT*(DP8(I,J) + DP(I,J))
      RETURN
      END
```

```
CARD
      SUBROUTINE SYSRUN
C ***
C     THIS ROUTINE CONTROLS THE CALCULATION OF THE MISSILE TRAJECTORY
C     AND TARGET-MISSILE INTERCEPT POINT. THE PRINT ROUTINE IS CALLED
C     AS REQUIRED TO PRINT RESULTS.
C ***
      COMMON / INCEPT/ UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON /STATEV/NT,UB,VB,WB,X(3),DUE(6)
      COMMON /COEFS/THR,AERC(18)
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON / SEEKR/ NS,BTHTG,BPSIG,OSV(10)
      COMMON / VANES/ NV,VV(4),DVV(4),DELQ,DELR,DELP
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCK1/P(33,33),DP(33,33)
      COMMON /BLOCK2/ A2(33,33),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON /BLOCK9/KOK
      COMMON /VMG/ H,MS
      COMMON /VMG9/JUNK,VTIME1,VTIME2,VNOISD,NUMM
      COMMON /MVMG/S3(40),KINTER,KCNTER
      COMMON /MVMG2/TEPSTG(33),KIT,IKPR,TMVE,TMRNG ,EZTMP,EYTMP
      COMMON /MBLOK1/KOUNT1,XNORM(4),S1(33,40)
      DOUBLE PRECISION T,DT,SVDT
      DIMENSION XMOLD(3),TOLD(3),XST(3)
C     ****************************
      IF(KIT.NE.0)GO TO 4
C     ****************************
C
C *** PRINT DATA HEADING AND INITIALIZE LAUNCHER DYNAMICS INDEX
C
      CALL PRHEAD
      LAUNCH = 1
C
C *** INITIALIZE AERODYNAMICS ROUTINE, DERIVATIVES AND TARGET POSITION.
C
      DELQ = 0.0
      DELR = 0.0
      DELP =0.0
      THR = 0.0
      T = 0.D0
      BEPSZ =0.
      BEPSY = 0.
      CALL THRCON
      CALL TRANSM
      CALL ROTATM
      CALL INTGT
      BEPSZ = 0.
      BEPSY = 0.
      CALL INSEEK
      CALL AUTOPT
      CALL VANEMD
      J = 1
      K=1
      DO 5 I=1,3
```

```
    5 XST(I) = X(I)
      SVDT = DT
      N = IDINT(DT/.5D-3)
      IPR = N*IPR
      DT = .5D-3
      CALL INSYST
      CALL INRK4
C     ****************************
    4 IF(IPR.EQ.40)K=1
C     ****************************
C
C *** INTEGRATE MISSILE EQUATIONS AND CALCULATE TARGET-MISSILE POSITION.
C
   10 KSTEP =0
      CALL PRDATA
   20 DO 25 I=1,3
      XMOLD(I)= X(I)
   25 TOLD(I) = XT(I)
      CALL SYSINT
      CALL TARGET
      CALL INSEEK
      GO TO (70,90),J
   70 IF(THR)80,80,90
   80 J=2
      CALL ROTZER
   90 GO TO (75,85,95),LAUNCH
   75 DO 76 I=1,3
   76 XMOLD(I) = X(I)-XST(I)
      CALL TRANS(ECSBCS,XMOLD,TOLD)
      IF (TOLD(1).LT.RL1)GO TO 45
      LAUNCH = 2
      WRITE( 6,910)T
      GO TO 45
   85 DO 86 I=1,3
   86 XMOLD(I) = X(I)-XST(I)
      CALL TRANS(ECSBCS,XMOLD,TOLD)
      IF(TOLD(1).LT.RL2)GO TO 45
      LAUNCH = 3
      WRITE( 6,920)T
      IPR = IPR/N
      N = IDINT(T/SVDT)+1
      DT = DFLOAT(N)*SVDT-T
      CALL INSYST
      CALL INRK4
      CALL SYSINT
      DT = SVDT
      KSTEP = MOD(N,IPR)-1
      CALL INSYST
      CALL INRK4
   95 GO TO (30,40),K
C
C *** IF MISSILE WITHIN 5 FT. OF TARGET,DIVIDE STEP LENGTH BY 2(FIRST TIME).
   30 IF(THRNGE.GT.5.)GO TO 40
      DT = .5D+0*DT
```

```
      IPR = IPR+IPR
      K = 2
C
C *** IF MISSILE-TARGET RELATIVE VELOCITY IS POSITIVE,INTERCEPT HAS
C *** OCCURRED
C
40    IF(TMVEL.GE.0.0) GO TO 50
      IF(T.GT.TSTOP) RETURN
45    KSTEP = KSTEP+1
C
C ****************
      IF(KONTER.NE.NUMM)GO TO 1
      GO TO 2
1     KONTER = KINTER
      IF(T.GT.VTIME1)RETURN
C ****************
C
2     IF (KSTEP-IPR)20,10,10
C
C *** CALCULATE MISS DISTANCE FROM CURRENT AND PREVIOUS POSITIONS
C
50    A = 0.
      B = 0.
      C = 0.
      DO 60 I=1,3
      TMP1 = XMOLD(I)-TOLD(I)
      A = A+TMP1*TMP1
      TMP2 = X(I)-XT(I)
      B = B+TMP2*TMP2
      TMP1 = X(I)-XMOLD(I)
60    C = C+TMP1*TMP1
      A=SQRT(A)
      B=SQRT(B)
      C = SQRT(C)
      Z = .5*(A+B+C)
      A = 2.*SQRT(Z*(Z-A)*(Z-B)*(Z-C))/C
      WRITE (  6,900) A
      WRITE(6,124)T
      IF(VTIME2.LT.TSTOP)GO TO 61
      DO 288 I=1,MS
288   WRITE(6,11)I,(P(I,K),K=1,I)
11    FORMAT(//1X,'P(',I2,', J) =',7E15.5/4(11X,7E15.5/))
124   FORMAT(1X,'TIME = ',F8.4)
 900  FORMAT (//20X,'*****  MISS DISTANCE   *****',F10.2, ' FT.')
 910  FORMAT (10X,'FIRST LUG OFF LAUNCHER AT T = ', F8.4)
 920  FORMAT (10X,'SECOND LUG OFF LAUNCHER AT T = ',F8.4)
61    RETURN
      END
```

```
      SUBROUTINE COEFF
C ***
C     THIS SUBROUTINE CALCULATES THE IMPLICIT 'A' MATRIX ELEMENTS
C ***
      COMMON / SEEKR/NS,BTHTG,BPSIG,BTHD,BPSD,EZ,EY,DSV(6)
      COMMON / INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON / AUTOP/NA,ZP1,ZP2,ZP3,ZY1,ZY2,ZY3,ZR1,ZR2,BPHIS,ZPI1,ZPI2,
     1EODCR,ZYI1,ZYI2,EVNCR,ZPD1,ZPD2,ZPD3,ZYD1,ZYD2,ZYD3,ZRD1,ZRD2,
     2BPHISD,ZPID1,ZPID2,EODCRD,ZYID1,ZYID2,EVNCRD,EZSS,EYSS,WQC,WRC,
     3EZRR,EYRR,BDELPC
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /STATEV/NT,UE,VE,WE,X(3),DUE,DVE,DWE,DX,DY,DZ
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WF,WQ,WR,BTHETA,BPH,BPS
      COMMON /MSINCG/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /VANEK/VGAIN,VLIM,VRLIM
      COMMON /COEFS/THR,CMQ,CNR,CNP,CY2,CL3,CXO,CMO,COCM,CNF,CN2,
     1CLP,CL2,CXC,CNQ,CMDQP,CLDRP,CMR,CLD
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VA,RHO
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /UTILTY/G,RTD
      COMMON /VMG/ H,MS
      COMMON /BLOCK1/P(33,33),DP(33,33)
      COMMON /BLOCK2/ A2(33,33),KIK,KOUNT,KICK,KAT
      COMMON /BLOCK6/ BACS(3)
      COMMON / BLOCK7/KK3,THRP,TIMP
      COMMON /BLOCK8/KK1,KK5,VP
      COMMON /BLOCK9/KOK
      COMMON /BLOCC1/CUE1,DVE1,DWE1,DPB1,DQB1,DRB1
      DOUBLE PRECISION T, DT
      DIMENSION X6(3),BCSEC1(3,3),ECSBC1(3,3),VV1(4),DEL1(3)
      KK1 = 0
      KK2 = 1
      KK3=7
      KK4 = 1
      KK6 = 1
      UE1 = UE
      PP1 = SQRT(ABS(P(16,16)))
      UE = UE + 0.1
      IF(PP1.GT.0.1 )  UE  =  UE1 + 0.1*PP1
      GO TO 143
643   DO 2 I = 1,3
2     DEL1(I) = DEL(I)
      IF(ABS(VV(II)) .LE.VLIM)GO TO 3
      VV(II) = XLIMIT(VV(II),VLIM)
3     TMP1 = VV(1)+VV(2)
      TMP2 = VV(3)+VV(4)
      DEL(1) = 0.25*(TMP1+TMP2)
```

```
      DEL(3) = 0.25*(TMP2-TMP1)
      DEL(2) = 0.25*(VV(2)+VV(4)-VV(1)-VV(3))
      KK1 = 0
      GO TO 343
  543 SNTHA1 = SNTHA
      CSTHA1 = CSTHA
      SNPHI1 = SNPHI
      CSPHI1 = CSPHI
      SNPSI1 = SNPSI
      CSPSI1 = CSPSI
      DO 7 IG=1,3
      DO 7 JJ=1,3
      BCSEC1(IG,JJ) = BCSECS(IG,JJ)
    7 ECSBC1(IG,JJ) = ECSBCS(IG,JJ)
  143 RHO1 = RHO
      VP1 = VP
      VA1 = VA
      QUE1 = QUE
      XMN1 = XMN
      ALFA1 = ALFA
      BETA1 = BETA
      ALFAP1 = ALFAP
      CSPHI1 = CSPHIP
      SNPHI1 = SNPHIP
      THRP1 = THRP
      TIMP1 = TIMP
      XM1 = XM
      XIX1 = XIX
      XIY1 = XIY
      RDCG1 = RDCG
      THR1 = THR
      CMQ1 = CMQ
      CNR1 = CNR
      CNP1 = CNP
      CY21 = CY2
      CL31 = CL3
      CX01 =CX0
      CM01 = CM0
      CDCM1 =CDCM
      CNF1 = CNF
      CN21 = CN2
      CLP1 = CLP
      CL21 = CL2
      CXC1 = CXC
      CNQ1 = CNQ
      CLDRP1 = CLDRP
      CMDQP1 = CMDQP
      CMR1 = CMR
      CLD1 = CLD
  343 FXA1 = FXA
      FYA1 = FYA
      FZA1 = FZA
      XMXA1 = XMXA
      XMYA1 = XMYA
```

```
      XMZA1 = XMZA
      FTHX1 = FTHX
      FTHY1 = FTHY
      FTHZ1 = FTHZ
      CALL TRANSM
      IF(KK2.EQ.2)GO TO 22
      CALL THRCON
      GO TO 22
144   A2(I1,J1) = (DUE1-DUE)/ZZ1
      A2(I1+1,J1) = (DVE1-DVE)/ZZ1
      A2(I1+2,J1) = (DWE1-DWE)/ZZ1
      A2(I1+6,J1)= (DPB1-DPB)/ZZ1
      A2(I1+7,J1)= (DQB1-DQB)/ZZ1
      A2(I1+8,J1)= (DRB1-DRB)/ZZ1
      IF(KK3.EQ.7)GO TO 155
      IF(KK1.EQ.1)GO TO 555
      DO 4 I = 1,3
4     DEL(I) = DEL1(I)
      GO TO 355
555   SNTHA = SNTHA1
      CSTHA  = CSTHA1
      SNPHI  = SNPHI1
      CSPHI  = CSPHI1
      SNPSI  = SNPSI1
      CSPSI  = CSPSI1
      DO 171 IG=1,3
      DO 171 JJ=1,3
      BCSECS(IG,JJ) = BCSEC1(IG,JJ)
171   ECSBCS(IG,JJ) = ECSBC1(IG,JJ)
155   RHO  = RHO1
      VP = VP1
      VA=VA1
      QUE = QUE1
      XMN  = XMN1
      ALFA  = ALFA1
      BETA  = BETA1
      ALFAP  = ALFAP1
      CSPHIP = CSPHI1
      SNPHIP = SNPHI1
      THRP  = THRP1
      TIMP  = TIMP1
      XM = XM1
      XIX = XIX1
      XIY = XIY1
      RDCG  = RDCG1
      THR = THR1
      CMQ = CMQ1
      CNR = CNR1
      CNP = CNP1
      CY2 = CY21
      CL3 = CL31
      CX0 = CX01
      CM0 = CM01
      CDCM = CDCM1
```

```
      CNF = CNF1
      CN2 = CN21
      CLP = CLP1
      CL2 = CL21
      CXC = CXC1
      CNQ= CNQ1
      CLDRP= CLDRP1
      CMDQP = CMDQP1
      CMR = CMR1
      CLD = CLD1
  355 FXA = FXA1
      FYA = FYA1
      FZA = FZA1
      XMXA = XMXA1
      XMYA = XMYA1
      XMZA= XMZA1
      FTHX  = FTHX1
      FTHY  = FTHY1
      FTHZ  = FTHZ1
      GO TO(143,543,643,343,57),KK6
   64 ZZ1 = UE -UE1
      KK4 = 2
      UE = UE1
      VE1 = VE
      PP1 = SQRT(ABS(P(17,17)))
      VE = VE + 0.001
      IF(PP1.GT..001)  VE  =  VE1 + 0.1*PP1
      I1 = 16
      J1 = 16
      GO TO 144
   44 ZZ1=VE-VE1
      KK4 = 3
      WE1 = WE
      VE = VE1
      PP1 = SQRT(ABS(P(18,18)))
      WE = WE + 0.1
      IF(PP1.GT.0.1 )  WE  =     . + 0.1*PP1
      J1 = 17
      GO TO 144
   45 ZZ1 = WE-WE1
      KK4 = 4
      X6(3) = X(3)
      WE=WE1
      PP1 = SQRT(ABS(P(21,21)))
      X(3) = X(3) + 1.0
      IF(PP1.GT.1.0 ) X(3) =X6(3) + 0.1*PP1
      J1 = 18
      GO TO 144
   46 ZZ1 = X(3)-X6(3)
      KK4 = 5
      THETA1 = THETA
      X(3) = X6(3)
      PP1 = SQRT(ABS(P(25,25)))
      THETA = THETA + 0.01
```

```
      IF(PP1.GT.0.01)THETA =THETA1+ 0.1*PP1
      KK1 = 1
      KK6 = 2
      J1 = 21
      GO TO 144
   47 ZZ1 = THETA-THETA1
      KK4 = 6
      THETA = THETA1
      PSI1 = PSI
      PP1 = SQRT(ABS(P(27,27)))
      PSI = PSI + 0.01
      IF(PP1.GT.0.01) PSI  = PSI1 + 0.1*PP1
      KK3 = 6
      J1 = 25
      GO TO 144
   48 ZZ1 = PSI-PSI1
      KK4 = 7
      PHI1 = PHI
      PSI = PSI1
      PP1 = SQRT(ABS(P(26,26)))
      PHI = PHI + 0.01
      IF(PP1.GT.0.01) PHI  = PHI1 + 0.1*PP1
      J1 = 27
      GO TO 144
   49 ZZ1 = PHI-PHI1
      KK4 = 8
      PHI = PHI1
      VV1(1) = VV(1)
      PP1 = SQRT(ABS(P(28,28)))
      VV(1) = VV(1) + 0.1
      IF(PP1.GT.0.1 ) VV(1)=VV1(1)+ 0.1*PP1
      KK5 = 1
      KK2 = 2
      KK6 = 3
      II = 1
      J1 = 26
      GO TO 144
   50 ZZ1 = VV(1)-VV1(1)
      KK4 = 9
      VV(1) = VV1(1)
      VV1(2) = VV(2)
      PP1 = SQRT(ABS(P(29,29)))
      VV(2) = VV(2) + 0.1
      IF(PP1.GT.0.1 ) VV(2)=VV1(2)+ 0.1*PP1
      II = 2
      J1 = 28
      GO TO 144
   51 ZZ1 = VV(2)-VV1(2)
      KK4 = 10
      VV(2) = VV1(2)
      VV1(3) = VV(3)
      PP1 = SQRT(ABS(P(30,30)))
      VV(3) = VV(3) + 0.1
      IF(PP1.GT.0.1 ) VV(3)=VV1(3)+ 0.1*PP1
```

```
          II = 3
          J1 = 29
          GO TO 144
   52     ZZ1 = VV(3)-VV1(3)
          KK4 = 11
          VV1(4) = VV(4)
          VV(3) = VV1(3)
          PP1 = SQRT(ABS(P(31,31)))
          VV(4) = VV(4) + 0.1
          IF(PP1.GT.0.1 ) VV(4)=VV1(4)+ 0.1*PP1
          II = 4
          J1 = 30
          GO TO 144
   53     ZZ1 = VV(4)-VV1(4)
          KK4 = 12
          PB1 = PB
          VV(4) = VV1(4)
          PP1 = SQRT(ABS(P(22,22)))
          PB = PB + 0.01
          IF(PP1.GT.0.01)  PB   =  PB1 + 0.1*PP1
          KK6 = 4
          WF1 = WF
          WF = PB*RTD
          J1 = 31
          GO TO 144
   54     ZZ1 = PB-PB1
          A2(22,22) = (DPB1-DPB)/ZZ1
          A2(23,22) = (DQB1-DQB)/ZZ1
          A2(24,22) = (DRB1-DRB)/ZZ1
          IF(LAUNCH.GT.2) GO TO 92
          A2(17,22) = (DVE1-DVE)/ZZ1
          A2(18,22) = (DWE1-DWE)/ZZ1
   92     KK4 = 13
          QB1 = QB
          PB = PB1
          WF = WF1
          WQ1 = WQ
          PP1 = SQRT(ABS(P(23,23)))
          QB = QB + 0.01
          IF(PP1.GT.0.01)  QB   =  QB1 + 0.1*PP1
          WQ = QB*RTD
          GO TO 355
   55     ZZ1 = QB - QB1
          A2(23,23) = (DQB1-DQB)/ZZ1
          A2(24,23) = (DRB1-DRB)/ZZ1
          IF(LAUNCH.GT,2) GO TO 93
          A2(17,23) = (DVE1-DVE)/ZZ1
          A2(18,23) = (DWE1-DWE)/ZZ1
   93     KK4 = 14
          RB1 = RB
          QB = QB1
          WQ = WQ1
          WR1 = WR
          PP1 = SQRT(ABS(P(24,24)))
```

```
      RB = RB + 0.01
      IF(PP1.GT.0.01)  RB  =  RB1 + 0.1*PP1
      WR = RB*RTD
      GO TO 355
56    ZZ1 = RB - RB1
      A2(23,24) = (DQB1-DQB)/ZZ1
      A2(24,24) = (DRB1-DRB)/ZZ1
      IF(LAUNCH.GT.2) GO TO 94
      A2(17,24) = (DVE1-DVE)/ZZ1
      A2(18,24) = (DWE1-DWE)/ZZ1
94    RB = RB1
      WR = WR1
      KK1 = 1
      KK3=0
      KK5 = 0
      KK6 = 5
      GO TO 355
22    DUE1 = BCSECS(1,1)*BACS(1)+BCSECS(1,2)*BACS(2)+BCSECS(1,3)*BACS(3)
      DVE1 = BCSECS(2,1)*BACS(1)+BCSECS(2,2)*BACS(2)+BCSECS(2,3)*BACS(3)
      DWE1 = BCSECS(3,1)*BACS(1)+BCSECS(3,2)*BACS(2)+BCSECS(3,3)*BACS(3)
     1+G
      GO TO (10,40),J
10    XMXTH = FTHZ*YTCG-FTHY*ZTCG
      XMYTH = ZTCG*FTHX+XTCG*FTHZ
      XMZTH = -YTCG*FTHX-XTCG*FTHY
40    XMX = XMXA+XMXTH
      XMY = XMYA+FZA*RDCG+XMYTH
      XMZ = XMZA-FYA*RDCG+XMZTH
      TMP1 = (1.-XIX/XIY)*PB
      DPB1 = XMX/XIX
      DQB1 = XMY/XIY+TMP1*RB
      DRB1 = XMZ/XIY-TMP1*QB
      GO TO(90,90,91),LAUNCH
90    CALL MDERIV
91    GO TO (64,44,45,46,47,48,49,50,51,52,53,54,55,56),KK4
57    IF(LAUNCH.GT.2) GO TO 95
      GO TO 96
95    IF(KOK.EQ.1) GO TO 96
      KOK = 1
      DO 97 I=17,18
      DO 97 I1=22,24
97    A2(I,I1) = 0.0
96    RETURN
      END
```

```
      SUBROUTINE MDERIV
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCC1/DUE1,DVE1,DWE1,DPB1,DQB1,DRB1
      DOUBLE PRECISION T,DT
      DIMENSION BACC(3)
      EQUIVALENCE (DVB,BACC(2)),(DWB,BACC(3))
      GO TO(30,50),LAUNCH
   30 RLCG1 = RLCG
      RLCG1 = RLCGO + RDCG
      CALL TRANS(ECSBCS,DUE1,BACC)
      TMP1 = RLCG1/XIY
      TMP2 = XM*RLCG1
      TMP3 = TMP1*TMP2 + 1.0
      FLY = (DRB1*TMP2-DVB*XM)/TMP3
      FLZ = -(DQB1*TMP2 + DWB*XM)/TMP3
      DVB = DVB + FLY/XM
      DWB = DWB + FLZ/XM
      DPB1 = 0.0
      DQB1 = DQB1 + FLZ*TMP1
      DRB1 = DRB1-FLY*TMP1
      CALL TRANS(BCSECS,BACC,DUE1)
      RETURN
   50 CALL TRANS(ECSBCS,DUE1,BACC)
      DVB = 0.0
      DWB = 0.0
      DPB1 = 0.0
      DQB1 = 0.0
      DRB1 = 0.0
      CALL TRANS(BCSECS,BACC,DUE1)
      RETURN
      END
```

```
CARD
      SUBROUTINE COVAR
      COMMON /VMG/ H,MS
      COMMON /VMG3/CONST1,CONST2
      COMMON /BLOCK1/P(33,33),DP(33,33),DP9(33,33)
      COMMON /BLOCK2/ A2(33,33),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      COMMON /TIMES/T,DT,TBC,TSTOP,IPR,J,LAUNCH
      DOUBLE PRECISION T,DT
      DIMENSION A3(15),P3(15)
      DO 25 I=1,MS
      DO 25 JJ=1,I
   25 P(JJ,I) = P(I,JJ)
      DO 201 I=1,15
      A3(I) = 0.0
  201 P3(I) = 0.0
      DO 1 II=1,MS
      DP(1,II) = A2(1,1)*P(1,II)+A2(1,2)*P(2,II)+A2(1,3)*P(3,II)
    1 DP(4,II) = A2(4,4)*P(4,II)+A2(4,5)*P(5,II)+A2(4,6)*P(6,II)
      DO 4 JI=1,MS
      DP(2,JI) = A2(2,1)*P(1,JI)
      DP(3,JI) = A2(3,2)*P(2,JI)
      DP(5,JI) = A2(5,4)*P(4,JI)
      DP(6,JI) = A2(6,5)*P(5,JI)
      DP(7,JI) = A2(7,7)*P(7,JI)+A2(7,8)*P(8,JI)+A2(7,26)*P(26,JI)
      DP(8,JI) = A2(8,7)*P(7,JI)
      DP(9,JI) = A2(9,7)*P(7,JI) + A2(9,8)*P(8,JI)+A2(9,26)*P(26,JI)
      DP(11,JI) = A2(11,10)*P(10,JI)
    4 DP(14,JI) = A2(14,13)*P(13,JI)
      DO 9 I = 10,12,2
      DO 9 JI = 1,MS
    9 DP(I,JI) = A2(I,2)*P(2,JI)+A2(I,3)*P(3,JI)+A2(I,5)*P(5,JI)+A2(I,6)
     1*P(6,JI)+A2(I,10)*P(10,JI)+A2(I,11)*P(11,JI)+A2(I,23)*P(23,JI)+
     2A2(I,24)*P(24,JI)
      DO 10 I=13,15,2
      DO 10 JI=1,MS
   10 DP(I,JI) = A2(I,2)*P(2,JI)+A2(I,3)*P(3,JI)+A2(I,5)*P(5,JI)+A2(I,6)
     1*P(6,JI)+A2(I,13)*P(13,JI)+A2(I,14)*P(14,JI)+A2(I,23)*P(23,JI)
     2+A2(I,24)*P(24,JI)
      JL = 16
      JM = 18
      KIT = 0
   17 DO 11 I=JL,JM
      DO 12 JK=1,3
   12 A3(JK) = A2(I,JK+15)
      A3(4) = A2(I,21)
      DO 13 JK=5,11
   13 A3(JK) = A2(I,JK+20)
      DO 11 II=1,MS
      DO 14 JK=1,3
   14 P3(JK) = P(JK+15,II)
      P3(4) = P(21,II)
      DO 15 JK=5,11
   15 P3(JK) = P(JK+20,II)
```

```
      DP(I,II) = 0.
      DO 11 JK=1,11
   11 DP(I,II) = DP(I,II) + A3(JK)*P3(JK)
      IF(KIT.EQ.1) GO TO 16
      KIT = 1
      JL = 22
      JM = 24
      GO TO 17
   16 DO 18 I=1,MS
   18 DP(22,I) = DP(22,I)+A2(22,22)*P(22,I)
      DO 19 JK=23,24
      DO 19 I=1,MS
   19 DP(JK,I) = DP(JK,I)+A2(JK,22)*P(22,I)+A2(JK,23)*P(23,I)+A2(JK,24)
     1*P(24,I)
      II = 16
      DO 20 JK=19,21
      DO 26 I=1,MS
   26 DP(JK,I) = A2(JK,II)*P(II,I)
      II=II+1
   20 CONTINUE
      DO 21 JK=25,27
      DO 21 I=1,MS
      DP(JK,I) = A2(JK,23)*P(23,I)+A2(JK,24)*P(24,I)+A2(JK,26)*P(26,I)
      DP(26,I) = DP(26,I)+A2(26,22)*P(22,I)+A2(26,25)*P(25,I)
   21 DP(27,I)  = DP(27,I) + A2(27,25)*P(25,I)
      JL = 28
      JI = 12
      DO 23 JK=28,31
      IF(JK.EQ.30)JI=12
      DO 27 I=1,MS
   27 DP(JK,I) = A2(JK,7)*P(7,I)+A2(JK,8)*P(8,I)+A2(JK,9)*P(9,I) +
     1A2(JK,26)*P(26,I)+A2(JK,JI)*P(JI,I)+A2(JK,JL)*P(JL,I)
      JL = JL+1
      JI = JI+3
   23 CONTINUE
      IF(LAUNCH.GT.2)GO TO 81
      DO 82 JK=17,18
      DO 82 I=1,MS
   82 DP(JK,I)=DP(JK,I)+A2(JK,22)*P(22,I) +A2(JK,23)*P(23,I) +A2(JK,24)*
     1P(24,I)
   81 DO 99 II=1,MS
      DO 99 JJ=1,MS
   99 DP9(JJ,II) = DP(II,JJ)
      DO 24 II=1,MS
      DO 24 JJ=1,II
   24 DP(II,JJ) =DP(II,JJ)+DP9(II,JJ)
      DP(1,1) = DP(1,1) + EZNOIS*B2(1)*B2(1)
      DP(4,4) = DP(4,4) + EYNOIS*B2(2)*B2(2)
      DP(28,28) = DP(28,28) + VGAIN*VGAIN*0.25
      DP(29,29) = DP(29,29) + VGAIN*VGAIN*0.25
      DP(30,28) = DP(30,28) + VGAIN*VGAIN*0.25
      DP(30,30) = DP(30,30) + VGAIN*VGAIN*0.25
      DP(31,29) = DP(31,29) + VGAIN*VGAIN*0.25
      DP(31,31) = DP(31,31) + VGAIN*VGAIN*0.25
      DP(32,32) = EZNOIS
      DP(33,32) = 0.0
      DP(33,33) = EYNOIS
      RETURN
      END
```

CARD

```
      SUBROUTINE SEEKER
      COMMON / SEEKR/NS,BTHTG,BPSIG,BTHD,BPSD,EZ,EY,OSVV(6)
      COMMON / SEEKK/SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON / INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /UTILTY/G,RTD
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS,VBEPS,VBEPSZ,VBEPSY
      COMMON /VNG9/JUNK,VTIME1,VTIME2,VNOISD,NUMM,NOMNAL
      DOUBLE PRECISION T,DT
      ENTRY INSEEK
      I = IDINT(T*1.D3+.5D0)
      I = MOD(I,50)
      IF(I.NE.0) RETURN
      TMP1 = TMRNGE/32810.
      TMP1 = .75*TMP1*TMP1
      EZ = DEAD(-TMP1,TMP1, BEPSZ)*SKSP
C ********************
      IF(NOMNAL.EQ.0)GO TO 1
      IF((T.LE.VTIME1).OR.(T.GE.VTIME2))GO TO 1
      VBEPS = VBEPSZ
      CALL SNOISE(TMP1,BEPSZ,EZ,EZNOIS)
1     EY = DEAD(-TMP1,TMP1,BEPSY)*SKSY
C ********************
      IF(NOMNAL.EQ.0)GO TO 2
      IF((T.LE.VTIME1).OR.(T.GE.VTIME2))GO TO 2
      VBEPS = VBEPSY
      CALL SNOISE(TMP1,BEPSY,EY,EYNOIS)
2     BTHTG = BTHTG + DTSAMP*EZ
      BPSIG = BPSIG + DTSAMP*EY
      RETURN
      END
```

CARD

```
      FUNCTION DEAD(P1,P2,X)
C
C     DEAD SPACE
C
      DEAD =0.0
      IF(X.GT.P1.AND.X.LT.P2)RETURN
      DEAD = SIGN(1.0,X)
      RETURN
      END
```

```
      SUBROUTINE SNOISE(TMP1,BEPS,EC,SGSQ)
      COMMON / SEEKK/SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS
      COMMON /UTILTY/G,RTD
      COMMON /STATEV/NT,UE,VE,WE,X(3)
      COMMON /BLOCK1/P(33,33),DP(33,33)
      COMMON /TIMES/T
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS,VBEPS
      COMMON /VMG9/JUNK,VTIME1,VTIME2,VNOISD
      DOUBLE PRECISION T
      SGTMP1 = (0.75/(32810.*32810.))*SQRT(10.0*P(19,19)*X(1)*X(1)
     1+3.0*P(19,19)*P(19,19)+10.0*P(20,20)*X(2)*X(2)
     2+3.0*P(20,20)*P(20,20) +10.0*P(21,21)*X(3)*X(3)
     3+3.0*P(21,21)*P(21,21) + 2.0*P(19,19)*P(20,20)
     4+2.0*P(19,19)*P(21,21) + 2.0*P(20,20)*P(21,21) +4.0E8*P(19,19)
     5-8.0E4*P(19,19)*X(1))
      SIGBEP = SQRT(VNOISD*VNOISD/(RTD*RTD)+SGTMP1*SGTMP1+VBEPS*VBEPS)
      IF(EC.NE.-SKSP) GO TO 21
      DIST = -TMP1 - BEPS
      POS = DIST/SIGBEP
      CALL DETARA (POS,AL1)
      AL = AL1+ 0.5
      POS = POS + 2.0*TMP1/SIGBEP
      CALL DETARA (POS,AO1)
      AO = AO1 - AL1
      AU = 1.0 - AL - AO
      GO TO 41
 21   IF(EC.NE.0.0) GO TO 22
      DIST = BEPS + TMP1
      POS = DIST/SIGBEP
      CALL DETARA (POS,AO1)
      AL = 0.5 - AO1
      POS =(TMP1 - BEPS)/SIGBEP
      CALL DETARA (POS,AO2)
      AU = 0.5 - AO2
      AO = AO1 + AO2
      GO TO 41
 22   DIST = BEPS - TMP1
      POS = DIST/SIGBEP
      CALL DETARA (POS,AU1)
      AU = AU1+ 0.5
      POS = POS + 2.0*TMP1/SIGBEP
      CALL DETARA(POS,AO1)
      AO = AO1 - AU1
      AL = 1.0 - AU - AO
 41   SIGEC =         AL*(-SKSP) + AU*SKSP
      SGSEC = (AU+AL)*SKSP*SKSP
      SGSQ = SGSEC - SIGEC*SIGEC
      WRITE(6,1)
 1    FORMAT(1X,'TIME',T21,'SIGBEP',T36,'EC',T51,'AL',T66,'AO',T81,
     1'AU',T96,'SIGEC',T111,'SGSQ'/)
      WRITE(6,2)T,SIGBEP,EC,AL,AO,AU,SIGEC,SGSQ
 2    FORMAT(1X,8E15.5)
      WRITE(6,3)DP(1,1),DP(4,4),BEPS
 3    FORMAT(1X,'DP(1,1) = ',E15.5,'DP(4,4) = ',E15.5,'BEPS = ',E15.5)
      RETURN
      END
```

```
      SUBROUTINE DETARA (POS,AAA)
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS
      III = 0
      DO 23 I=1,30
      AA1 = (I-1)/10.0
      AA2 = 0.1 + AA1
      III = III + 1
      IF(POS.GT.AA1.AND.POS.LE.AA2)GO TO 24
   23 CONTINUE
      III = 31
      CORRCT = 0.0
      GO TO 25
   24 CORRCT = 10.0*(POS-AA1)*(AREA(III)-AREA(III+1))
   25 AAA = 0.5-AREA(III) + CORRCT
      RETURN
      END
```

```
      SUBROUTINE VANEMD
C ***
C     THIS ROUTINE EVALUATES DERIVATIVES FOR INTEGRATION VARIABLES
C     USED IN THE VANES MODULE.
C ***
      COMMON / AUTOP/NA,ZP1,ZP2,ZP3,ZY1,ZY2,ZY3,ZR1,ZR2,BPHIS,ZPI1,ZPI2,
     1EODCR,ZYI1,ZYI2,EVNCR,ZPD1,ZPD2,ZPD3,ZYD1,ZYD2,ZYD3,ZRD1,ZRD2,
     2BPHISD,ZPID1,ZPID2,EODCRD,ZYID1,ZYID2,EVNCRD,EZSS,EYSS,WQC,WRC,
     3EZRR,EYRR,BDELPC
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /VANEK/VGAIN,VLIM,VRLIM
      COMMON /BLOCK4/ VV5(4),DLTC(4)
      COMMON /BLIK2/ AVD(4),BVD(4)
      COMMON /MBLOK1/KOUNT1,XNORM(4)
      DLTC(1) = EODCR+BDELPC +XNORM(1)*0.5
      DLTC(2) = EVNCR+BDELPC + XNORM(2)*0.5
      DLTC(3) = EODCR-BDELPC +XNORM(1)*0.5
      DLTC(4) = EVNCR-BDELPC + XNORM(2)*0.5
      DO 30 I=1,4
      VV5(I) = VV(I)
      IND = 1
      IF(ABS(VV(I)).LE.VLIM)GO TO 10
      IND = 2
      VV(I)= XLIMIT(VV(I),VLIM)
10    DVV(I) = XLIMIT(VGAIN*(DLTC(I)-VV(I)),VRLIM)
      GO TO(30,20),IND
      AVD(I) = DVV(I)
      BVD(I) = DVV(I)*VV(I)
20    IF(DVV(I)*VV(I).GT.0.)DVV(I)=0.
30    CONTINUE
      TMP1 = VV(1)+VV(2)
      TMP2 =VV(3)+VV(4)
      DEL(1) = 0.25*(TMP1+TMP2)
      DEL(3) = 0.25*(TMP2-TMP1)
      DEL(2) = 0.25*(VV(2)+VV(4)-VV(1)-VV(3))
      RETURN
      END
```

```
      SUBROUTINE TARGET
C
C *** THIS ROUTINE CALCULATES TARGET/MISSILE RELATIVE POSITION AND
C *** SPEED AND GENERATES LINE-OF-SIGHT SIGNALS IN SEEKER PLATFORM
C *** COORDINATES
C
      COMMON / SEEKR / NS,VS(2),DVS(2),OSV(8)
      COMMON /STATEV/NT,UE(3), X(3),DUE(3),DX(3)
      COMMON / INCEPT/ UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /UTILTY/G,RTD
      COMMON /TIMES/T
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS,VBEPS,VBEPSZ,VBEPSY
      COMMON /BLOCK1/P(33,33),DP(33,33)
      COMMON /MBLOK1/KOUNT1,XNORM(4)
      COMMON /VMG9/JUNK,VTIME1,VTIME2,VNOISD,NUMM,NOMNAL
      DIMENSION RMP(3),SMP(3),TMP(3)
      DOUBLE PRECISION T
      EQUIVALENCE (RXBA,RMP(1)),(RYBA,RMP(2)),(RZBA,RMP(3))
      EQUIVALENCE (RXG,SMP(1)),(RYG,SMP(2)),(RZG,SMP(3))
      A = 0.0
      B = 0.0
      C = 0.0
      DO 10 I=1,3
      SMP(I) = UT(I)-UE(I)
      TMP(I) = XT(I)-X(I)
      RMP(I) = TMP(I)-SMP(I)
      A = A+TMP(I)*TMP(I)
   10 B = B+SMP(I)*SMP(I)
      TMRNGE = SQRT(A)
      TMVEL = SQRT(B)
      COSA =0.
      DO 20 I=1,3
      A = TMP(I)/TMRNGE
      B =SMP(I)/TMVEL
   20 COSA = COSA+A*B
      TMVEL = COSA*TMVEL
      A = VS(1)/RTD
      CSTHG = COS(A)
      SNTHG = SIN(A)
      A = VS(2)/RTD
      CSPSG = COS(A)
      SNPSG = SIN(A)
      A = TMP(1)*CSTHG-TMP(3)*SNTHG
      RXG = A*CSPSG+TMP(2)*SNPSG
      RYG = TMP(2)*CSPSG - A*SNPSG
      RZG = TMP(3)*CSTHG + TMP(1)*SNTHG
      BEPSZ = ATAN(-RZG/RXG) +XNORM(3)*VNOISD/RTD
      BEPSY = ATAN(RYG/RXG)  +XNORM(4)*VNOISD/RTD
C ********************
      IF(NOMNAL.EQ.0)GO TO 1
      IF((T.LE.VTIME1).OR.(T.GE.VTIME2))GO TO 1
C ********************
      D = (1.+(-RZG/RXG)*(-RZG/RXG))**2
```

```
      E = (1.+( RYG/RXG)*( RYG/RXG))**2
      F = RXG**4
      H = 1.0/(D*F)
      H1= 1.0/(E*F)
      Q =(-TMP(1)*SNTHG-TMP(3)*CSTHG)*CSPSG
      Q1 = (RXG*(TMP(3)*SNTHG-TMP(1)*CSTHG)+RZG*Q)/RTD
      Q2 = RXG*SNTHG-RZG*CSTHG*CSPSG
      Q3 = -RZG*SNPSG
      Q4 = RXG*CSTHG+RZG*SNTHG*CSPSG
      Q5 =RZG*(-A      *SNPSG+TMP(2)*CSPSG)/RTD
      R1 = (RXG*(TMP(1)*SNTHG+TMP(3)*CSTHG)*SNPSG-RYG*Q)/RTD
      R2 = RXG*CSTHG*SNPSG+RYG*CSTHG*CSPSG
      R3 = -RXG*CSPSG+RYG*SNPSG
      R4 =-RXG*SNTHG*SNPSG-RYG*SNTHG*CSPSG
      R5 =(RXG* (-TMP(2)*SNPSG-A*CSPSG)-RYG*(-A*SNPSG+TMP(2)*CSPSG))/RTD
      VBEPSZ = H*(Q1*Q1*P(32,32)+Q2*Q2*P(19,19)+Q3*Q3*P(20,20)
     1+Q4*Q4*P(21,21)+Q5*Q5*P(33,33)+2.0*Q1*Q2*P(32,19)
     2+2.0*Q1*Q3*P(32,20)+2.0*Q2*Q4*P(21,19)+2.0*Q2*Q5*P(33,19)
     3+2.0*Q3*Q4*P(21,20)+2.0*Q3*Q5*P(33,20)+2.0*Q4*Q5*P(33,21)+
     42.0*Q1*Q5*P(33,32)+2.0*Q2*Q3*P(20,19)+2.0*Q1*Q4*P(32,21))
      VBEPSY =H1*(R1*R1*P(32,32)+R2*R2*P(19,19)+R3*R3*P(20,20)
     1+R4*R4*P(21,21)+R5*R5*P(33,33)+2.0*R1*R2*P(32,19)
     2+2.0*R1*R3*P(32,20)+2.0*R2*R4*P(21,19)+2.0*R2*R5*P(33,19)
     3+2.0*R3*R4*P(21,20)+2.0*R3*R5*P(33,20)+2.0*R4*R5*P(33,21)+
     42.0*R1*R5*P(33,32)+2.0*R2*R3*P(20,19)+2.0*R1*R4*P(32,21))
    1 RETURN
      ENTRY INTGT
      VS(1) = ATAN((X(3)-XT(3))/XT(1))*RTD
      VS(2) = 0.
      RETURN
      END
```

```
      SUBROUTINE ROTATM
C ***
C     THIS ROUINE CALCULATES DERIVATIVES FOR THE MISSILE ROTATIONAL
C     VARIABLES PB,QB,RB AND THE EULER ANGLES THETA, PHI, PSI.
C ***
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      DOUBLE PRECISION T,DT
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /UTILTY/G,RTD
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      DIMENSION BACC(3)
      EQUIVALENCE (DVB,BACC(2)),(DWB,BACC(3))
C
C *** MOMENTS DUE TO THRUST MISALIGNMENT
C
      GO TO (10,40),J
10    XMXTH = FTHZ*YTCG-FTHY*ZTCG
      XMYTH = ZTCG*FTHX+XTCG*FTHZ
      XMZTH = -YTCG*FTHX-XTCG*FTHY
C
C *** TOTAL APPLIED MOMENTS
C
40    XMX = XMXA+XMXTH
      XMY = XMYA+FZA*RDCG+XMYTH
      XMZ = XMZA-FYA*RDCG+XMZTH
C
C *** DERIVATIVES
C
      TMP1 = (1.-XIX/XIY)*PB
      DPB = XMX/XIX
      DQB = XMY/XIY+TMP1*RB
      DRB = XMZ/XIY-TMP1*QB
      DTHA = QB*CSPHI-RB*SNPHI
      DPSI = (RB*CSPHI+QB*SNPHI)/CSTHA
      DPHI = PB+DPSI*SNTHA
      WP = PB*RTD
      WQ = QB*RTD
      WR = RB*RTD
      BPH = PHI*RTD
C
C *** MODIFY DERIVATIVES WHEN LAUNCHER DYNAMICS ARE IN EFFECT
C
      GO TO (50,30,20),LAUNCH
20    RETURN
30    RLCG = RLCGO+RDCG
      CALL TRANS(ECSBCS,DUE,BACC)
      TMP1= RLCG/XIY
      TMP2 = XM*RLCG
```

CARD

```
      TMP3 = TMP1*TMP2+1.
      FLY = (DRB*TMP2-DVB*XM)/TMP3
      FLZ = -(DQB*TMP2+DWB*XM)/TMP3
      DVB= DVB+FLY/XM
      DWB = DWB+FLZ/XM
      DPB =0.
      DQB = DQB+FLZ*TMP1
      DRB = DRB-FLY*TMP1
      CALL TRANS(BCSECS,BACC,DUE)
      RETURN
   50 CALL TRANS(ECSBCS,DUE,BACC)
      DVB = 0.
      DWB =0.
      DPB =0.
      DQB =0.
      DRB = 0.
      CALL TRANS(BCSECS,BACC,DUE)
      RETURN
      ENTRY ROTZER
      XMXTH =0.
      XMYTH = 0.
      XMZTH  =0.
      RETURN
      END
```

CARD

```
      SUBROUTINE TRANS(TMTX,VECTOR,RESULT)
      DIMENSION TMTX(3,3),VECTOR(3),RESULT(3)
      RESULT(1) = TMTX(1,1)*VECTOR(1)+TMTX(1,2)*VECTOR(2)+TMTX(1,3)*
     1VECTOR(3)
      RESULT(2) = TMTX(2,1)*VECTOR(1)+TMTX(2,2)*VECTOR(2)+TMTX(2,3)*
     1VECTOR(3)
      RESULT(3) = TMTX(3,1)*VECTOR(1)+TMTX(3,2)*VECTOR(2)+TMTX(3,3)*
     1VECTOR(3)
      RETURN
      END
```

```
      SUBROUTINE TRANSM
C ***
C     THIS ROUTINE CALCULATES DERIVATIVES FOR THE TRANSLATIONAL
C     EQUATIONS OF MISSILE MOTION, INCLUDING LAUNCHER DYNAMICS WHEN
C     APPROPRIATE.
C ***
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WF,WQ,WR,BTHETA,BPH,BPS
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /MSINCG/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCK6/ BACS(3)
      COMMON /COEFS/THR,AERC(18)
      COMMON /UTILTY/G,RTD
      COMMON / BLOCK7/KK3,THRP,TIMP
      COMMON /BLOCK8/KK1,KK5,VP
      DIMENSION ANGLS(6)
      EQUIVALENCE (ANGLS(1),PB)
C
C *** CALCULATE EULER TRIGONOMETRICAL TERMS
C
      IF(KK1.EQ.0)GO TO 20
      SNTHA = SIN(THETA)
      CSTHA = COS(THETA)
      SNPHI = SIN(PHI)
      CSPHI = COS(PHI)
      SNPSI = SIN(PSI)
      CSPSI = COS(PSI)
C
C *** CALCULATE BODY/EARTH AND EARTH/BODY TRANSFORMATION MATRICES
C
      TMP1 = SNPHI*SNTHA
      TMP2 = CSPHI*SNTHA
      BCSECS(1,1) = CSPSI*CSTHA
      BCSECS(2,1) = SNPSI*CSTHA
      BCSECS(3,1) =-SNTHA
      BCSECS(1,2)= CSPSI*TMP1-SNPSI*CSPHI
      BCSECS(2,2)= SNPSI*TMP1+CSPSI*CSPHI
      BCSECS(3,2)= CSTHA*SNPHI
      BCSECS(1,3)= CSPSI*TMP2+SNPSI*SNPHI
      BCSECS(2,3)= SNPSI*TMP2-CSPSI*SNPHI
      BCSECS(3,3)= CSTHA*CSPHI
      DO 15 I=1,3
      DO 15 K=1,3
15    ECSBCS(I,K)= BCSECS(K,I)
C
C *** CALCULATE AERODYNAMIC FORCES AND MOMENTS
C
20    CALL AERODY
C
C *** CALCULATE THRUST COMPONENTS
```

```
C
      FTHX = THR*COSAT
      FTHY = THR*SATPHI
      FTHZ = THR*SATCPH
C
C *** CALCULATE BODY ACCELERATIONS EXCLUDING GRAVITY
C
      BACS(1) = (FTHX-FXA)/XM
      BACS(2) = (FTHY+FYA)/XM
      BACS(3) = (FTHZ+FZA)/XM
      IF(KK3.NE.0)RETURN
C
C *** TRANSFORM BODY ACCELERATIONS TO ECS AND CALCULATE DERIVATIVES
C
      CALL TRANS(BCSECS,BACS,DUE)
      DWE = DWE+G
      DX = UE
      DY = VE
      DZ = WE
      RETURN
      ENTRY INTRAN
C
C *** CALCULATE THRUST ANGLES AS SINES AND COSINES
C
      TMP1 = SQRT(XTCG*XTCG+YTCG*YTCG+ZTCG*ZTCG)
      COSAT = XTCG/TMP1
      SATPHI = YTCG/TMP1
      SATCPH = ZTCG/TMP1
C
C *** CONVERT INITIAL VALUES TO RADIANS
C
      DO 10 I=1,6
   10 ANGLS(I) = ANGLS(I)/RTD
      RETURN
      END
```

```
      SUBROUTINE AUTOPT
      COMMON / AUTOP/NA,ZP1,ZP2,ZP3,ZY1,ZY2,ZY3,ZR1,ZR2,BPHIS,ZPI1,ZPI2,
     1EODCR,ZYI1,ZYI2,EVNCR,ZPD1,ZPD2,ZPD3,ZYD1,ZYD2,ZYD3,ZRD1,ZRD2,
     2BPHISD,ZPID1,ZPID2,EODCRD,ZYID1,ZYID2,EVNCRD,EZSS,EYSS,WQC,WRC,
     3EZRR,EYRR,BDELPC
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /SEEKR/ NS,VS(2),DVS(2),OSV(8)
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /BLIK1/BPHISM
      EQUIVALENCE  (EZ,OSV(1)),  (EY,OSV(2))
C *** LIMITATION OF INTEGRATORS*
      EODCR = XLIMIT(EODCR,PYLIM)
      EVNCR = XLIMIT(EVNCR,PYLIM)
C *** GUIDANCE FILTER - PITCH
      ZPD1 = GYZ*EZ-TAUZ*((3.*(ZP1+TAUZ*ZP2))+TAUZ*TAUZ*ZP3)
      ZPD2 = ZP1
      ZPD3 = ZP2
      EZSS = TAUL*ZP3+ZP2
C *** GUIDANCE FILTER - YAW
      ZYD1 = GYZ*EY-TAUY*((3.*(ZY1+TAUY*ZY2))+TAUY*TAUY*ZY3)
      ZYD2 = ZY1
      ZYD  -  Y2
      EYS     AUL*ZY3+ZY2
      WQC = _ZSS+QBIAS+GBIAS
      WRC = EYSS + RBIAS
      WQDIF = WQ -WQC
      WRDIF = WR -WRC
      EZRR = WQDIF-WRDIF
      EYRR = WQDIF+WRDIF
C *** ROLL COMPENSATION
      ZRD1 = WP1*(WP1*(BPH-ZR2)-2.*DP1*ZR1)
      ZRD2 = ZR1
      BPHISM = RK1*(ZR2+((RA1+RB2)*ZR1+ZRD1)/RA1/RB2)
      BPHISD = XLIMIT(BPHISM,RLIM)
      BDELPC =0.1*(BPHIS + 10.0*BPHISD)
C *** PITCH INTEGRATOR
      ZPID1 = TMP7*EZRR - TMP2*ZPI1 - TMP1*ZPI2
      ZPID2 = ZPI1
      EODCRD = TMP3*ZPI2+TMP4*ZPI1+ZPID1
C *** YAW INTEGRATOR*
      ZYID1 = TMP7*EYRR - TMP2*ZYI1 - TMP1*ZYI2
      ZYID2 = ZYI1
      EVNCRD = TMP3*ZYI2+TMP4*ZYI1+ZYID1
      RETURN
      ENTRY INAUPT
      TMP1 = WQ1*WQ1
      TMP2 = 2.*DQ1*WQ1
      TMP3 = PYAK1*PYBK1
      TMP4 = PYAK1+PYBK1
      TMP5 = WQG*WQG
      TMP6 = 2.*DQG*WQG
      TMP7 = PYIK1*WQ1*WQ1/TMP3
      RETURN
      END
```

```
      SUBROUTINE AERODY
C ***
C     THIS ROUTINE EVALUATES AERODYNAMIC FORCES AND MOMENTS APPLIED TO
C     THE MISSILE, USING COEFFICENTS AND DERIVATIVES OBTAINED BY TABLE
C     INTERPOLATION. FORCES AND MOMENTS ARE RETURNED IN COMMON BLOCK
C     /FCEMOM/.
C ***
      COMMON /COEFS/THR,CMQ,CNR,CNP,CY2,CL3,CXO,CMO,CDCM,CNF,CN2,
     1CLP,CL2,CXC,CNQ,CMDQP,CLDRP,CMR,CLD
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VA,RHO
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /VANES/NV,VVD(8),DELQ,DELR,DELP
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCK8/KK1,KK5,VP
      DOUBLE PRECISION T, DT
      DIMENSION BVEL(3),DUM(3)
      EQUIVALENCE  (UB,BVEL(1)),(VB,BVEL(2)),(WB,BVEL(3))
      IF(KK5.EQ.1)GO TO 30
      DUM(1) = UE-WUE
      DUM(2) = VE-WVE
      DUM(3) = WE-WWE
      CALL TRANS(ECSBCS,DUM,BVEL)
      RHO = 2.3738E-3+6.7844E-8*Z
      VA = 1116.08+3.6292E-3*Z
      TMP1 = VB*VB+WB*WB
      VP = UB*UB+TMP1
      TMP1 = SQRT(TMP1)
      QUE = 0.5*RHO*VP
      VP = SQRT(VP)
      XMN=VP/VA
      ALFA = ATAN(WB/UB)
      BETA = ATAN(VB/UB)
      ALFAP = ATAN(TMP1/UB)
      IF (TMP1.EQ.0.)GO TO 40
      CSPHIP = WB/TMP1
      SNPHIP = VB/TMP1
      GO TO 50
 40   CSPHIP = 1.
      SNPHIP = 0.
 50   CONTINUE
      GO TO (10,20),J
 10   CALL DTLUX1
      GO TO 30
 20   CALL DTLUX2
 30   SN2PHI = 2.*SNPHIP*CSPHIP
      SN4PHI = 2.*SN2PHI*(CSPHIP-SNPHIP)*(CSPHIP+SNPHIP)
      SN2PHI = SN2PHI*SN2PHI
      TMP1 = DELR*CMR
      TMP2 = DELQ*CMDQP
```

```
      TMP3 = TMP1*CSPHIP+TMP2*SNPHIP
      TMP4 = TMP2*CSPHIP-TMP1*SNPHIP
      TMP1 = CNP*SN4PHI+TMP3
      TMP2 = CMO+CDCM*SN2PHI+TMP4
      CM = CSPHIP*TMP2+SNPHIP*TMP1
      CN = CSPHIP*TMP1-SNPHIP*TMP2
      CL = CL2*SN4PHI+CL3*SNPHIP+DELP*CLD
      CX=CXO+CXC
      TMP1 = DELR*CLDRP
      TMP2 = DELQ*CNQ
      TMP3 = TMP1*CSPHIP+TMP2*SNPHIP
      TMP4 = TMP2*CSPHIP-TMP1*SNPHIP
      TMP1 = CY2*SN4PHI+TMP3
      TMP2 = CNF+CN2*SN2PHI+TMP4
      CY = CSPHIP*TMP1-SNPHIP*TMP2
      CZ = -CSPHIP*TMP2-SNPHIP*TMP1
      TMP1 = QUE*S
      FXA = TMP1*CX
      FYA = TMP1*CY
      FZA = TMP1*CZ
      TMP1 = TMP1*D
      TMP2 = 0.5*D/VP
      XMXA = TMP1*(CL+WP*TMP2*CLP)
      XMYA= TMP1*(CM+WQ*TMP2*CMQ)
      XMZA = TMP1*(CN+WR*TMP2*CNR)
      RETURN
      END
```

```
      SUBROUTINE DTLUX1
C ***
C     THIS ROUTINE OBTAINS THRUST AND AERODYNAMIC COEFFICIENTS AND
C     DERIVATIVES FROM TABLE INTERPOLATION.  TABULATED FUNCTIONS ARE
C     HELD IN BLANK COMMON AND ROUTINE INTRP3 IS CALLED TO PERFORM THE
C     ACTUAL INTERPOLATION. RESULTS ARE RETURNED IN COMMON BLOCK /COEFS/
C ***
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VSS,RHO
      COMMON /TIMES/T
      COMMON /VANES/SKP1(9),DELQ,DELR,DELP
      COMMON /COEFS/THR,CMQ,CNR,CNP,CY2,CL3,CXO,CMO,CDCM,CNF,CN2,CLP,CL2
     1,CXC,CNQ,CMDQP,CLDRP,CMR,CLD
      COMMON /UTILTY/G,RTD
      DOUBLE PRECISION T
      DIMENSION ONEDM(4), TWODM(7)
      EQUIVALENCE (ONEDM(1),CNP),(TWODM(1),CMO)
      T1 = SNGL(T)
      IF(T1.GT..14)GO TO 10
      CALL INTRP3(T1,0.,0.,1,THR)
      GO TO 20
   10 CALL INTRP3(T1,0.,0.,2,THR)
      GO TO 20
      ENTRY DTLUX2
20    ALF = ABS(ALFA)*RTD
      BET = ABS(BETA)*RTD
      ALFP = ALFAP*RTD
      DQ = ABS(DELQ)
      DR = ABS(DELR)
      CALL INTRP3(ALF,0.,0.,3,CMQ)
      CALL INTRP3(BET,0.,0.,3,CNR)
      DO 30 I=4,6
   30 CALL INTRP3(ALFP,0.,0.,I,ONEDM(I-3))
      CALL INTRP3(XMN,0.,0.,7,CXO)
      DO 40 I=8,14
   40 CALL INTRP3(ALFP,XMN,0.,I,TWODM(I-7))
      CALL INTRP3(ALFP,XMN,DQ,15,CNQ)
      CALL INTRP3(ALFP,XMN,DR,15,CLDRP)
      CALL INTRP3(ALFP,XMN,DQ,16,CMDQP)
      CALL INTRP3(ALFP,XMN,DR,16,CMR)
      CALL INTRP3(ALFP,XMN,ABS(DELP),17,CLD)
      RETURN
      END
```

```
      SUBROUTINE THRCON
C ***
C     THIS ROUTINE CALCULATES MISSILE MASS, INERTIAS AND CG POSITION
C     AS A FUNCTION OF ENGINE THRUST CONDITIONS. THE INTEGRAL OF THE
C     THRUST IS CALCULATED BY THE TRAPEZOIDAL RULE TO OBTAIN ENGINE
C     IMPULSE.
C ***
      COMMON /COEFS/THR,AERC(18)
      COMMON /MSINCG/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /UTILTY/G,RTD
      COMMON / BLOCK7/KK3,THPP,TIMP
      DOUBLE PRECISION T,DT
      TIMP = TIMP+.5*(T-TPR)*(THR+THRP)
      THRP = THR
      TPR = T
      DELW = TIMP/SI
      XM = (WO-DELW)/G
      TMP1 = 1.-DELW/WO
      XIX = XIXO*TMP1
      XIY = XIYO*TMP1
      RDCG = RDCGO-DELW*CGSHWP
      RETURN
      ENTRY INTHRC
C
C *** ZERO STARTING VALUES OF THRUST INTEGRAL AND TIME
C
      TIMP = 0.
      TPR =0.
      THRP = 0.
      CGSHWP = (RDCGO-RDCGP)/WP
      RETURN
      END
```

```
      SUBROUTINE INTRP3(X,Y,Z,I,FXYZ)
C ***
C     THIS ROUTINE PERFORMS LINEAR INTERPOLATION IN TABULATED FUNCTIONS
C     OF 1, 2 OR 3 INDEPENDENT VARIABLES. THE FUNCTIONS MUST BE
C     TABULATED FOR VALUES OF INDEPENDENT VARIABLES WHICH START AT ZERO
C     AND INCREASE WITH CONSTANT INTERVALS. THE TABLES USED ARE DEFINED
C     FOR POSITIVE RANGES OF INDEPENDENT VARIABLES BUT IF REQUIRED
C     THE VARIABLE INCREMENT MAY BE NEGATIVE.
C ***
      COMMON DXDYDZ(60),IADD(20),AERO(1360)
      J = 3*I-2
      DX = DXDYDZ(J)
      DY = DXDYDZ(J+1)
      DZ = DXDYDZ(J+2)
      J = IFIX(X/DX)
      DELX = X/DX-FLOAT(J)
      IF (DY.EQ.0.)GO TO 40
      IF (J.GT.16)J=16
      K = IFIX(Y/DY)
      DELY = Y/DY-FLOAT(K)
      IF (K.GT.4)K=4
      IF (DZ.EQ.0.)GO TO 50
      L = IFIX(Z/DZ)
      DELZ = Z/DZ-FLOAT(L)
      IF (L.GT.4)L=4
      M = J+16*K+64*L+IADD(I)
      N = 1
      NN = 2
      GO TO 30
   10 M = M+64
      N = 2
      FXY1 = FXY
      GO TO 30
   20 FXYZ = FXY1+(FXY-FXY1)*DELZ
      RETURN
   40 M = J+IADD(I)
      NN = 1
      GO TO 30
   50 M = J+16*K+IADD(I)
      NN = 2
      N = 3
      GO TO 30
   60 FXYZ = FX1
      RETURN
   70 FXYZ = FXY
      RETURN
   30 FX1 = AERO(M)+(AERO(M+1)-AERO(M))*DELX
      GO TO(60,80),NN
   80 M = M+16
      FX2 = AERO(M)+(AERO(M+1)-AERO(M))*DELX
      M = M-16
      FXY = FX1+(FX2-FX1)*DELY
      GO TO(10,20,70),N
      END
```

```
      SUBROUTINE PROATA
      COMMON /SEEKR/ NS,VS(2),DVS(2),OSV(8)
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      DOUBLE PRECISION T,DT
      COMMON /CNTRL/DUM(6),DATA(64)
      COMMON /AUTOP/NA,VA(15),DVA(15),OV(7)
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VSS,RHO
      COMMON /COEFS/THR,AERC(18)
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON / INCEPT/ UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /UTILTY/G,RTD
      DIMENSION RDRV(6), DDRV(6)
      EQUIVALENCE (RDRV(1),DPB)
      BTHETA = THETA*RTD
      BPS = PSI*RTD
      GO TO(40,50,60), ISW
   40 RETURN
   50 WRITE(  6,930) T,UE,VE,WE,X,Y,Z,WP,WQ,WR,BTHETA,BPH,BPS,UT,XT,
     1 TMRNGE,TMVEL,VS
      LINES = LINES+3
      IF(LINES .LT. 52) RETURN
      LINES = 1
      IPAGE = IPAGE+1
      WRITE (  6,940) IPAGE
      RETURN
   60 CONTINUE
      CONTINUE
      IPAGE = IPAGE+1
      WRITE (  6,940) IPAGE
      ALFAP = ALFAP*RTD
      ALFA = ALFA*RTD
      BETA = BETA*RTD
      CSPHIP = ATAN2(SNPHIP,CSPHIP)*RTD
      DO 70 I=1,6
   70 DDRV(I) = RDRV(I)*RTD
      WRITE(  6,950) T,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      WRITE(  6,960) WP,WQ,WR,BTHETA,BPH,BPS,DDRV
      WRITE(  6,970) VS,DVS
      WRITE(  6,980) VA,DVA
      WRITE(  6,990) VV,DVV
      WRITE(  6,1000) DEL,BEPSZ,BEPSY,OSV,OV
      WRITE(  6,1010) XMN,VSS,RHO,QUE,ALFAP,ALFA,BETA,CSPHIP,AERC,
     1 FXA,FYA,FZA,XMXA,XMYA,XMZA
      WRITE(  6,1020) FTHX,FTHY,FTHZ,XM,XIX,XIY,RDCG
      WRITE(  6,1030) UT,XT,TMRNGE,TMVEL
```

CARD

```
      RETURN
      ENTRY PRHEAD
      WRITE( 6,900) (DATA(I),I=1,20)
      WRITE( 6,920) S,D,RL1,RL2,WO,WF,XIXO,XIYO,RDCGO,RDCGP,QBIAS,
     1RBIAS,XTCG,YTCG,ZTCG,WUE,WVE,WWE,RLCGO,SI,DT
      LINES = 40
      IPAGE = 1
      IF (IPR)10,20,30
10    ISW = 3
      IPR = -IPR
      RETURN
20    ISW = 1
      RETURN
30    ISW = 2
      WRITE( 6,910)
      RETURN
900   FORMAT(1H1,120X,'PAGE 1',/48X,'TERMINAL HOMING SIMULATION (DIGITAL
     1', /48X,36('-'), //20X,20A4 //)
910   FORMAT(//25X,'RESULTS ROW 1:', /30X,'COLUMN 1 TIME IN SECONDS',
     125X,'COLUMN 2 UE IN FT/SEC', /30X,'COLUMN 3 VE IN FT/SEC',28X,
     2'COLUMN 4 WE IN FT/SEC', /30X,'COLUMN 5 MISSILE X COORD IN FT',
     319X,'COLUMN 6 MISSILE Y COORD IN FT', /30X,'COLUMN 7 MISSILE Z
     4COORD IN FT', 19X,'COLUMN 8 ROLL RATE IN DEG/SEC', /30X,'COLUMN 9
     5 PITCH RATE IN DEG/SEC', 19X,'COLUMN 10 YAW RATE IN DEG/SEC', /30
     6X,'COLUMN 11 THETA IN DEGREES', 24X,'COLUMN 12 PHI IN DEGREES', /
     730X,'COLUMN 13 PSI IN DEGREES', //25X,'RESULTS ROW 2:', /30X,
     8'COLUMN 2 TARGET U IN FT/SEC', 21X,'COLUMN 3 TARGET V IN FT/SEC'
     9, /30X,'COLUMN 4 TARGET W IN FT/SEC', 21X,'COLUMN 5 TARGET X COORD
     A IN FT',/30X,'COLUMN 6 TARGET Y COORD IN FT', 19X,'COLUMN  7 TA
     BRGET Z COORD IN FT', /30X,'COLUMN 8 MISSILE/TARGET RANGE IN FT',
     C13X,'COLUMN 9 MISSILE/TARGET CLOSING SPEED IN FT/SEC', /30X,'COLU
     DMN 10 GIMBAL ANGLE THETAG IN DEGREES', 9X,'COLUMN 11 GIMBAL ANGLE
     EPSIG IN DEGREES')
920   FORMAT (5X,'VEHICLE DETAILS:', //10X,'REFERENCE AREA', 15X,F8.3,
     1' SQ FT', 20X,'REFERENCE LENGTH', 12X,F8.3, ' FT', /10X,'FRONT LUG
     2 LAUNCHER TRAVEL',4X,F8.3,'FT',23X,'REAR LUG LAUNCHER TRAVEL', 4X,
     3F8.3, ' FT',/10X,'INITIAL TOTAL WEIGHT', 9X,F8.2, ' LBS', 22X,
     4'PROPELLANT WEIGHT', 10X,F8.2, ' LBS',/10X,'INITIAL X MOM. OF I.',
     5 9X,F8.3, ' SLUGS FT**2', 14X,'INITIAL Y MOM. OF I.',  8X,F8.3,
     6' SLUGS FT**2', /10X,'CG TOTAL SHIFT',15X,F8.3, ' FT', 23X,
     7'PROPELLANT CG TO CGO', 8X,F8.3, ' FT', /10X,'AUTOPILOT Q BIAS',
     813X,F8.3, ' DEG/SEC', 18X,'AUTOPILOT R BIAS', 12X,F8.3, ' DEG/SEC'
     9/10X,'THRUST POINT OFFSETS (X,Y,Z FT)',10X,3F10.2,/10X,'WIND SPEED
     A COMPONENTS (XE,YE,ZE F/S)', 5X,3F10.1, /10X,'REAR LUG TO CGO(FT)'
     B,22X,F10.3,/10X,'ENGINE SPECIFIC IMPULSE', 6X,F8.3, ' SECS', 21X,
     C    'INTEGRATION STEP LENGTH', 5X,F8.4, ' SECS')
930   FORMAT (/3X,F6.3, 2(3F10.2, 3F10.1),/9X,3F10.2, 4F10.1, 3F10.2)
940   FORMAT( 1H1,30X,'TERMINAL HOMING CONTD ....', 51X,'PAGE', I3)
950   FORMAT (// 10X,'TIME',F8.3, ' SECONDS', //5X,'TRANSLATION VARIAB
     1LES IN F/SEC AND FT', 12X,3F10.2, 3F10.1, /5X,'TRANSLATION DERIVAT
     2IVES IN F/SEC**2 AND F/SEC', 5X,3F10.3, 3F10.2)
960   FORMAT (/5X,'ROTATION VARIABLES IN DEG/SEC AND DEGS', 11X,6F10.2,
     1/5X,'ROTATION DERIVATIVES IN DEG/SEC**2 AND DEG/SEC', 4X,6F10.3)
970   FORMAT (/5X,'SEEKER VARIABLES IN DEG AND DEG/SEC', 15X,2F10.3,/5X,
```

```
      1 'SEEKER DERIVATIVES IN DEG/SEC AND DEG/SEC**2',  8X,2F10.3)
 980  FORMAT (/5X,'AUTOPILOT VARIABLES IN DEG ETC', 20X,6F10.3, /55X,
      1 6F10.3, /55X,7F10.3, /5X,'AUTOPILOT DERIVATIVES IN DEG ETC', 18X,
      26F10.3, /55X,6F10.3, /55X,7F10.3)
 990  FORMAT (/5X,'VANE VARIABLES IN DEGREES', 25X,4F10.3, /5X,
      1 'VANE DERIVATIVES IN DEG/SEC', 23X,4F10.3)
1000  FORMAT (/5X,'DELQ,DELR, DELP(DEGREES)',11X,3F8.3,11X,'BEPSZ & BEP
      1SY(DEGS)', 2X,2F8.3,//5X,'SEEKER ADDITIONAL VARIABLES', 4X,8F10.3
      2,//5X,'AUTOPILOT ADDITIONAL VARIABLES', 10X,7F10.3)
1010  FORMAT (/5X,'MACH NO', F9.2, 4X,'SONIC SP', F8.1, 4X,'AIR DENS',
      2F8.6, 4X,'DYN PRES', F8.2, 4X,'ALFA P', F10.3, 4X,'ALFA', F12.3,
      2/5X 'BETA', F12.3, 4X,'PHI PR', F10.3,//5X,'AERODYNAMIC COEFFICIENT
      3TS', /5X,'CMO(A)', F10.4, 4X,'CNR(B)', F10.4, 4X,'CNP(A)', F10.4,
      44X,'CY2(A)', F10.4, 4X,'CL3(A)', F10.4, 4X,'CAO(M)', F10.4, 4X/5X,
      5'CMO(A,M)', F8.4, 4X,'CDCM(A,M)', F7.4, 4X,'CNF(A,M)', F8.4, 4X,
      6'CN2(A,M)', F8.4, 4X,'CLP(A,M)', F8.4, 4X,'CL2(A,M)', F8.4, /5X,
      7'CXC(A,M)', F8.4, 4X,'CNQ(A,M,Q)', F6.4, 4X,'CMDQP(3V)', F7.4, 4X,
      8'CLDRP(3V)', F7.4,4X,'CMR(A,M,R)', F6.4, 4X,'CLC(A,M,P)', F6.4,
      9// 5X,'AERODYNAMIC FORCES AND MOMENTS',/5X,'FXA(LB)', F9.2, 4X,
      A'FYA(LB)', F9.2, 4X,'FZA(LB)', F9.2, 4X,'MXA(LBFT)', F7.2, 4X,
      B'MYA(LBFT)', F7.2, 4X,'MZA(LBFT)', F7.2)
1020  FORMAT (/5X,'THRUST COMPONENTS (X,Y,Z LB)', 3F8.1, 4X,'MASS', F8.2
      1, 4X,'X M. OF I.', F8.2, 4X,'Y M. OF I.', F8.3,/5X,'CG SHIFT',20X,
      2 F8.3)
1030  FORMAT (/5X,'TARGET SPEED (X,Y,Z FT/SEC)', 3F8.1, 4X,'TARGET POSIT
      1ION (X,Y,Z FT)',3F10.1,/5X,'TARGET/MISSILE RANGE (FT)', F10.1,20X,
      2 'CLOSING SPEED (F/S)', 9X,F8.1)
      END
```

CARD										
1	3.	3.	0.	0.05	0.	0.	0.	0.		
2	373.	1.	15.	15.	2.	6750.	12.	60.		
3	130.	.53	.33	40.	15.	2.8	115.	.64		
4	15.	7.	1.	0.	0.	0.	0.	0.		
5	.1	0.	0.	5.	0.	0.	0.	0.		
6	-40.	.267	.584	2.75	0.	0.	3.5	6.07		
7	0.	0.	0.	195.8	121.	19.4	.241	15.11		
8	2.54	-.375	-.15	15.	20.	200.				
9	0.50	0.4602	0.4207	0.3821	0.3446	0.3085	0.2743	0.2420	0.2119	0.1861
10	0.1587	0.1357	0.1151	0.0968	0.0808	0.0668	0.0548	0.0446	0.0359	0.0287
11	0.0228	0.0179	0.0139	0.0107	0.00820	0.00621	0.00466	0.00347	0.00256	0.00187
12	8	.02			THRUST TABLE 1 FOR TIME 0 TO .14 SECS					
13	0.5	2850.	2660.	2240.	2230.	2205.	2180.	2170.		
14	48	.1			THRUST TABLE 2 FOR TIME FROM .14 SECS					
15	0.5	2205.	2160.	2140.	2125.	2110.	2095.	2075.		
16	2060.	2040.	2020.	2005.	1990.	1970.	1950.	1910.		
17	1800.	1200.	610.	420.	320.	295.	220.	190.		
18	140.	120.	100.	90.	80.	75.	65.	55.		
19	48.	41.	35.	30.	20.	10.	0.			
20										
21	16	2.			TABLE OF RATE DAMPING DERIVATIVS CMQ					
22	-4.1	-5.25	-6.3	-7.4	-8.4	-9.3	-9.96	-10.45		
23	-10.78	-10.95	-11.0	-11.0	-11.0	-11.0	-11.0	-11.0		
24	15	2.			DELTA CN PRIME					
25	0.	.05	.18	.4	.69	1.06	1.5	2.01		
26	2.59	3.22	3.86	4.73	4.73	4.73	4.73	4.73		
27	16	2.			DELTA CY PRIME					
28	0.	-.015	-.07	-.17	-.3	-.47	-.65	-.87		
29	-1.1	-1.345	-1.6	-1.86	-1.86	-1.86	-1.86	-1.86		
30	16	2.			DELTA CL PRIME LUGS					
31	0.	.015	.025	.032	.045	.051	.08	.11		
32	.145	.181	.215	.255	.255	.255	.255	.255		
33	16	.0916667			CXO PRIME					
34	.465	.445	.43	.411	.397	.387	.379	.375		
35	.420	.558	.730	.970	1.2	1.2	1.2	1.2		
36	16 4	2.	.366667		CMO PRIME					
37	0.	-.95	-2.1	-3.6	-5.2	-7.2	-9.3	-11.55		
38	-13.8	-16.2	-18.55	-21.1	-21.1	-21.1	-21.1	-21.1		
39	0.	-.95	-2.1	-3.6	-5.2	-7.2	-9.3	-11.55		
40	-13.8	-16.2	-18.55	-21.1	-21.1	-21.1	-21.1	-21.1		
41	0.	-.95	-2.1	-3.6	-5.2	-7.2	-9.3	-11.55		
42	-13.8	-16.2	-18.55	-21.1	-21.1	-21.1	-21.1	-21.1		
43	0.	-.6	-1.6	-3.1	-4.75	-6.7	-8.8	-10.95		
44	-13.2	-15.5	-17.8	-20.2	-20.2	-20.2	-20.2	-20.2		
45	16 4	2.	.366667		DELTA CM PRIME					
46	0.	-.03	-.14	-.3	-.64	-1.19	-1.85	-2.63		
47	-3.46	-4.36	-5.38	-6.45	-6.45	-5.45	-6.45	-6.45		
48	0.	-.03	-.14	-.3	-.64	-1.19	-1.85	-2.63		
49	-3.46	-4.36	-5.38	-6.45	-6.45	-6.45	-6.45	-6.45		
50	0.	-.03	-.14	-.3	-.64	-1.19	-1.85	-2.63		
51	-3.46	-4.36	-5.38	-6.45	-6.45	-6.45	-6.45	-6.45		
52	0.	-.05	-.17	-.4	-.75	-1.32	-2.02	-2.8		
53	-3.68	-4.6	-5.65	-6.8	-6.8	-6.8	-6.8	-6.8		
54	16 4	2.	.366667		CN PRIME					

CARD								
55	0.	.69	1.4	2.2	3.15	4.24	5.38	6.54
56	7.72	9.04	10.55	12.2	12.2	12.2	12.2	12.2
57	0.	.69	1.4	2.2	3.15	4.24	5.38	6.54
58	7.72	9.04	10.55	12.2	12.2	12.2	12.2	12.2
59	0.	.69	1.4	2.2	3.15	4.24	5.38	6.54
60	7.72	9.04	10.55	12.2	12.2	12.2	12.2	12.2
61	0.	.69	1.4	2.2	3.2	4.35	5.5	6.74
62	8.0	9.37	10.7	12.0	12.	12.	12.	12.
63	16 4	2.	.366667		DELTA CN PRIME			
64	0.	.015	.06	.155	.31	.5	.75	1.05
65	1.395	1.78	2.2	2.63	2.63	2.63	2.63	2.63
66	0.	.015	.06	.155	.31	.5	.75	1.05
67	1.395	1.78	2.2	2.63	2.63	2.63	2.63	2.63
68	0.	.015	.06	.155	.31	.5	.75	1.05
69	1.395	1.78	2.2	2.63	2.63	2.63	2.63	2.63
70	0.	.015	.06	.155	.32	.53	.8	1.11
71	1.46	1.84	2.26	2.71	2.71	2.71	2.71	2.71
72	16 4	2.	.366667		ROLL DAMPING CLP			
73	-.232	-.315	-.39	-.464	-.527	-.579	-.62	-.649
74	-.668	-.675	-.67	-.645	-.645	-.645	-.645	-.645
75	-.232	-.315	-.39	-.464	-.527	-.579	-.62	-.649
76	-.668	-.675	-.67	-.645	-.645	-.645	-.645	-.645
77	-.232	-.315	-.39	-.464	-.527	-.579	-.62	-.649
78	-.668	-.675	-.67	-.645	-.645	-.645	-.645	-.645
79	-.25	-.333	-.41	-.482	-.55	-.609	-.657	-.698
80	-.728	-.75	-.72	-.72	-.72	-.72	-.72	-.72
81	16 4	2.	.366667		DELTA CLP PRIME			
82	0.	.007	.02	.045	.07	.101	.122	.193
83	.25	.297	.331	.354	.354	.354	.354	.354
84	0.	.007	.02	.045	.07	.101	.122	.193
85	.25	.297	.331	.354	.354	.354	.354	.354
86	0.	.007	.02	.045	.07	.101	.122	.193
87	.25	.297	.331	.354	.354	.354	.354	.354
88	0.	.008	.035	.07	.12	.186	.277	.387
89	.515	.672	.84	1.03	1.03	1.03	1.03	1.03
90	15 4	2.	.366667		CXC			
91	0.	0.	0.	0.	.002	.02	.055	.13
92	.24	.387	.642	1.09	1.09	1.09	1.09	1.09
93	0.	0.	0.	0.	.002	.02.	.055	.13
94	.24	.387	.642	1.09	1.09	1.09	1.09	1.09
95	0.	0.	0.	0.	.002	.02	.055	.13
96	.24	.387	.642	1.09	1.09	1.09	1.09	1.09
97	0.	0.	0.	0.	.002	.008	.026	.07
98	.135	.23	.365	.56	.56	.56	.56	.56
99	16 4 4	2.	.366667	10.	CN PRIME PER DELTA R OR Q			
100	.143	.1425	.145	.151	.157	.162	.166	.1735
101	.182	.1867	.1895	.191	.191	.191	.191	.191
102	.143	.1425	.145	.151	.157	.162	.166	.1735
103	.182	.1867	.1895	.191	.191	.191	.191	.191
104	.143	.1425	.145	.151	.157	.162	.166	.1735
105	.182	.1867	.1895	.191	.191	.191	.191	.191
106	.179	.1795	.1825	.188	.196	.203	.210	.217
107	.227	.231	.232	.232	.232	.232	.232	.232
108	.143	.1425	.145	.151	.157	.162	.166	.1735

CARD								
109	.182	.1867	.1895	.191	.191	.191	.191	.191
110	.143	.1425	.145	.151	.157	.162	.166	.1735
111	.182	.1867	.1895	.191	.191	.191	.191	.191
112	.143	.1425	.145	.151	.157	.162	.166	.1735
113	.182	.1867	.1895	.191	.191	.191	.191	.191
114	.179	.1795	.1825	.188	.196	.203	.210	.217
115	.227	.231	.232	.232	.232	.232	.232	.232
116	.175	.169	.171	.176	.184	.192	.201	.2095
117	.216	.219	.22	.22	.22	.22	.22	.22
118	.175	.169	.171	.176	.184	.192	.201	.2095
119	.216	.219	.22	.22	.22	.22	.22	.22
120	.175	.169	.171	.176	.184	.192	.201	.2095
121	.216	.219	.22	.22	.22	.22	.22	.22
122	.205	.204	.205	.209	.214	.22	.226	.233
123	.24	.247	.254	.262	.262	.262	.262	.262
124	.175	.169	.171	.176	.184	.192	.201	.2095
125	.216	.219	.22	.22	.22	.22	.22	.22
126	.175	.169	.171	.176	.184	.192	.201	.2095
127	.216	.219	.22	.22	.22	.22	.22	.22
128	.175	.169	.171	.176	.184	.192	.201	.2095
129	.216	.219	.22	.22	.22	.22	.22	.22
130	.205	.204	.205	.209	.214	.22	.226	.233
131	.24	.247	.254	.262	.262	.262	.262	.262
132	16 4 4	2.	.366667	10.	CM PRIME	PER DELTA	R OR Q	
133	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
134	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
135	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
136	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
137	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
138	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
139	-.76	-.75	-.753	-.771	-.8	-.83	-.857	-.886
140	-.917	-.95	-.98	-1.01	-1.01	-1.01	-1.01	-1.01
141	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
142	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
143	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
144	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
145	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
146	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
147	-.76	-.75	-.753	-.771	-.8	[illegible]	-.857	-.886
148	-.917	-.95	-.98	-1.01	-1.01	[illegible]	-1.01	-1.01
149	-.795	-.783	-.786	-.795	-.01	[illegible]	-.862	-.898
150	-.922	-.935	-.93	-.9	-.9	[illegible]	-.9	-.9
151	-.795	-.783	-.786	-.795	-.81	[illegible]	-.862	-.898
152	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
153	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
154	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
155	-.865	-.84	-.83	-.848	-.87	-.893	-.92	-.94
156	-.965	-.994	-1.02	-1.05	-1.05	-1.05	-1.05	-1.05
157	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
158	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
159	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
160	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
161	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
162	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9

CARD								
163	-.865	-.84	-.83	-.848	-.87	-.893	-.92	-.94
164	-.965	-.994	-1.02	-1.05	-1.05	-1.05	-1.05	-1.05
165	16 4 4	2.	.366667	10.	CL PRIME PER DELTA P			
166	.13	.127	.125	.124	.123	.122	.1225	.124
167	.124	.123	.12	.116	.116	.116	.116	.116
168	.13	.127	.125	.124	.123	.122	.1225	.124
169	.124	.123	.12	.116	.116	.116	.116	.116
170	.13	.127	.125	.124	.123	.122	.1225	.124
171	.124	.123	.12	.116	.116	.116	.116	.116
172	.143	.14	.1375	.135	.133	.131	.13	.129
173	.128	.1285	.13	.132	.132	.132	.132	.132
174	.13	.127	.125	.124	.123	.122	.1225	.124
175	.124	.123	.12	.116	.116	.116	.116	.116
176	.13	.127	.125	.124	.123	.122	.1225	.124
177	.124	.123	.12	.116	.116	.116	.116	.116
178	.13	.127	.125	.124	.123	.122	.1225	.124
179	.124	.123	.12	.116	.116	.116	.116	.116
180	.143	.14	.1375	.135	.133	.131	.13	.129
181	.128	.1285	.13	.132	.132	.132	.132	.132
182	.142	.1455	.146	.144	.14	.138	.137	.136
183	.1355	.1345	.134	.134	.134	.134	.134	.134
184	.142	.1455	.146	.144	.14	.138	.137	.136
185	.1355	.1345	.134	.134	.134	.134	.134	.134
186	.142	.1455	.146	.144	.14	.138	.137	.136
187	.1355	.1345	.134	.134	.134	.134	.134	.134
188	.148	.146	.144	.142	.14	.139	.138	.137
189	.136	.136	.1355	.135	.135	.135	.135	.135
190	.142	.1455	.146	.144	.14	.138	.137	.136
191	.1355	.1345	.134	.134	.134	.134	.134	.134
192	.142	.1455	.146	.144	.14	.138	.137	.136
193	.1355	.1345	.134	.134	.134	.134	.134	.134
194	.142	.1455	.146	.144	.14	.138	.137	.136
195	.1355	.1345	.134	.134	.134	.134	.134	.134
196	.148	.146	.144	.142	.14	.139	.138	.137
197	.136	.136	.1355	.135	.135	.135	.135	.135
198	999							
199	1							
200	TOTAL SYSTEM CHECKOUT RUN FOR DR J. ROWLAND, 7 APRIL 1972.							
201	.0025	15.0		40				